Lehrstuhl Angewandte Informatik IV
Datenbanken und Informationssysteme
Prof. Dr.-Ing. Stefan Jablonski

Institut für Angewandte Informatik
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth

Seminar

Vorname Nachname

August 26, 2015
Version: Draft / Final

Universität Bayreuth

Fakultät Mathematik, Physik, Informatik
Institut für Informatik
Lehrstuhl für Angewandte Informatik IV

Titel / Topic

Seminar

Vorname Nachname

1. Reviewer Prof. Dr.-Ing. Stefan Jablonski
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

2. Reviewer Dr. Stefan Schönig
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

Supervisors Stefan Schönig and Lars Ackermann

August 26, 2015

Vorname Nachname
Seminar
Titel / Topic, August 26, 2015
Reviewers: Prof. Dr.-Ing. Stefan Jablonski and Dr. Stefan Schönig
Supervisors: Stefan Schönig and Lars Ackermann

Universität Bayreuth
Lehrstuhl für Angewandte Informatik IV
Institut für Informatik
Fakultät Mathematik, Physik, Informatik
Universitätsstrasse 30
95447 Bayreuth
Germany

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Abstract (different language)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Acknowledgement

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction

1.1 Motivation

Machine learning (ML) has become a pivotal tool in decision-making processes across numerous domains, including healthcare, finance and hiring. However, as these models increasingly influence critical decisions, questions about their fairness and the ethical implications of their use have come to the forefront. Fairness in ML concerns the equitable treatment of individuals or groups, particularly in the presence of sensitive attributes such as gender, age, race, or socioeconomic status. These attributes often correlate with historical inequalities or systemic biases embedded in the data. ML models, designed to optimize accuracy, learn patterns from this data and may inadvertently exploit these unfair patterns to make predictions. While leveraging such patterns can improve predictive performance, it also risks perpetuating or even amplifying existing inequities. A notable example of this occurred when Amazon's AI recruiting tool, which was designed to assist in hiring decisions, exhibited significant gender bias, favoring male candidates due to biases in historical hiring data, leading to its eventual scrapping. [Cha23]

These challenges of fairness and bias also extend to the field of predictive business process monitoring (PBPM). Here, ML models play an integral role in making decisions, whether by predicting outcomes, allocating resources, or streamlining operations. Addressing fairness in this context requires careful consideration of bias's dual nature. While biases can lead to discrimination, not all bias is inherently harmful. In some cases, certain biases may be necessary for the model to achieve its intended purpose. For instance, sensitive attributes like gender can carry significant information relevant to specific contexts. In the domain of healthcare, gender differences in biological and hormonal factors are crucial for determining effective treatments and drug prescriptions. However, the same attribute might lead to discriminatory outcomes in unrelated contexts, such as predicting whether a patient's request for treatment will be approved.

A significant obstacle in addressing this issue is the inherent opacity of ML models. Many models, especially complex ones such as deep neural networks, produce predictions without providing insight into how those predictions are made. This lack of interpretability makes it challenging for stakeholders and domain experts

to identify whether a model's reliance on a sensitive attribute aligns with ethical guidelines. This presents a dilemma: either leaving potentially harmful biases unchecked to avoid significantly compromising the model's predictive performance, or removing the influence of the sensitive attributes entirely from the model. The latter approach, while intended to prevent discrimination, can lead to suboptimal predictions, especially when the sensitive attributes carry critical information relevant to the task.

1.2 Problem Statement

To address this challenge, this thesis proposes an approach based on knowledge distillation. Knowledge distillation involves transferring the encapsulated knowledge from a complex model to a simpler, more interpretable representation. By applying this technique, the inner workings of a predictive ML model can be visualized and understood by human stakeholders and domain experts, which in turn enables the identification of inherent biases in the model. Specifically for the proposed approach, the distilled representation is modified to remove the unwanted bias by adjusting the way sensitive attributes influence the representation's decision-making process.

The goal of this approach is twofold: to make the ML model fairer by eliminating the biases that result in unfair treatment and to maintain its predictive accuracy by preserving helpful ones. By balancing these objectives, the proposed methodology seeks to create models that are not only effective but also aligned with given ethical standards and societal expectations.

1.3 Thesis Outline

Building on the problem statement, the second chapter provides insight into existing research concerning the field of bias reduction and fairness in PBPM, while highlighting the gaps we aim to address. The third chapter provides background information essential for understanding the thesis. The fourth chapter details our employed methodology. The fifth chapter presents the evaluation of the proposed approach, including empirical results from multiple case studies. The final chapter concludes the thesis by summarizing the findings and discussing their implications for fairness in ML. It acknowledges the limitations of the research and offers suggestions for future work.

2 Related Work

In recent years, the field of fairness in machine learning has gained significant attention, aiming to address biases and ensure equitable outcomes in predictive systems. Numerous approaches to quantifying and achieving fairness have been proposed, targeting various steps in the ML pipeline. These methods are too plentiful to go over in this thesis, but an overview can be found in [CH24].

Comparatively, the existing literature related to fairness specifically in PBPM is quite limited, although many foundational concepts and challenges concerning fairness can be transferred from general ML, as proposed in [DAFST22]. Accordingly, we will focus on reviewing approaches that have explicitly been adapted into the field of PBPM.

[QA19] represents one of the initial efforts to incorporate fairness into PBPM. This approach utilizes discrimination-aware decision trees, which extend traditional decision trees by introducing fairness constraints. Specifically, when the model's predictions exceed a certain threshold of unfairness, i.e. when they are disproportionately influenced by sensitive attributes like race or gender, a relabeling technique is applied. This relabeling adjusts the predictions of the model in a way, such that the discriminatory impact of sensitive attributes is reduced, while keeping the loss in predictive accuracy as small as possible. Although this technique is similar to our methodology as described in section 4.6, there is a key difference: The relabeling in this method is performed automatically, based on the assumption that all bias is undesirable and should be eliminated regardless of the context. In contrast, our approach emphasizes the involvement of domain expert to decide which biases are unwanted, allowing for the preservation of nuanced, context-dependent correlations.

In [LP24], biases in predictive models are being adressed by leveraging adversarial learning [Goo+14], a method where two models are trained simultaneously: a predictor and an adversary. The predictor model focuses on accurate outcomes, while an adversary model tries to identify sensitive attributes based on the output of the predictor. The predictor is penalized when the adversary succeeds, encouraging fairness by reducing reliance on these attributes. While effective at reducing bias, this penalty-based approach lacks context sensitivity, treating all biases as inherently undesirable. This indiscriminate removal of biases may unintentionally eliminate

meaningful correlations, thereby diminishing the predictive model's utility in specific scenarios where such relationships are crucial.

Lastly, [PDV24] focuses on ensuring independence between predictions and sensitive group memberships using a composite loss function for training the ML model. Compared to traditional loss functions, which seek to only optimize the predictive performance of a ML model, this composite loss function incorporates integral probability metrics [PZ19], in order to find a balance between accuracy and fairness. However, similar to the other approaches presented before, the uniform elimination of biases may inadvertently remove desirable correlations, potentially affecting the utility of the predictive model in certain contexts.

Work	Predictive Model	Fairness Metrics	Datasets
[QA19]	Decision Trees [Cor+09]	Demographic Parity [Dwo+12]	Hospital Billing Log [Man17]
[LP24]	Fully-connected Neural Networks [HRW86], LSTM [CDGR19]	Shapley Values [Sha53], Equalized Odds [HPS16]	BPI Challenge 2013 [Don12], Simulated Logs [Poh+23]
[PDV24]	LSTM [CDGR19]	Demographic Parity [Dwo+12], ABPC and ABCC [Han+23]	Simulated Logs [Poh+23]

Tab. 2.1: Comparison of related work, highlighting different utilized predictive models, fairness metrics, and datasets.

In conclusion, current approaches to fairness in PBPM have made significant strides but remain somewhat limited by their automated and uniform treatment of bias. A shift toward an approach that puts more control into the hands of domain experts presents a promising direction for creating more context-sensitive models, that achieve better performance while still making context-based fair decisions.

3 Background

This chapter will go over definitions and background knowledge necessary for understanding the methodology and evaluation presented in later chapters. It will adress fundamentals of Business Process Management, go over the basics of neural networks and decision trees, and explain how fairness is quantified in this thesis.

3.1 Business Process Management

Business processes represent structured sets of activities or tasks undertaken by organizations to achieve specific objectives, such as processing customer orders, managing inventory or onboarding new employees. A specific execution or instance of the overall business process is refered to as a **case**. For example, in an order fulfillment process, each order corresponds to a separate case.

3.1.1 Business Process Model and Notation

In order to provide visualization for stakeholders, business processes are often portrayed in the form of **process models**. These capture the sequence and flow in a formalized representation and are generally created using the standardized **Business Process Model and Notation (BPMN)** [Whi04], which we will also use a simplified subset of:

- **Activities** are represented as rectangles with rounded corners, denoting the individual tasks or operations within the process.

- **Gateways** are represented as diamond shapes with a cross inside and are used to manage the branching and merging of process flows. They serve as decision points where the workflow can split into different paths or converge back into a single path. In this thesis, we focus only on exclusive decision points, meaning that only one of the available paths can be taken at any given time. The probability of choosing each path will be explicitly annotated.

- **Events** are shown as circles, representing the initiation and completion of the process. Specifically, the start event, depicted as a circle with a narrow border, marks the beginning of the process and the end event, depicted as a circle with a bold border, signifies the process's conclusion.

- **Sequence Flow** is represented by lines with solid arrowheads and is used to show the order that activities will be performed in.

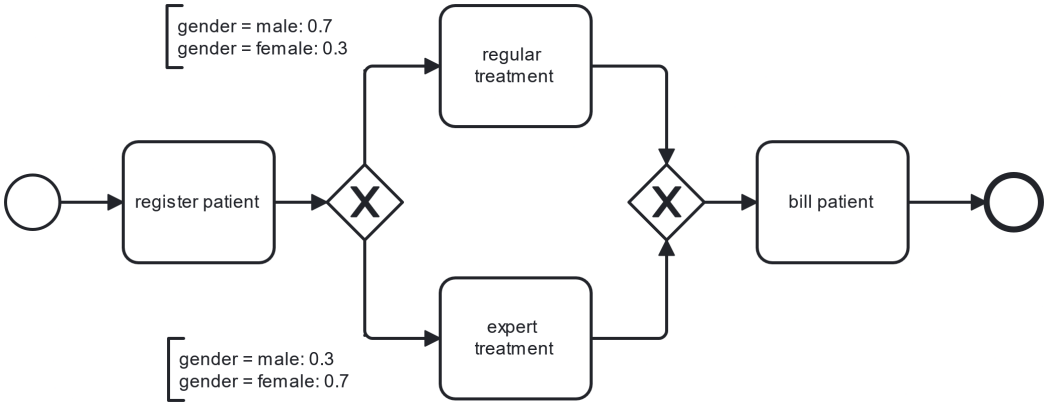

Fig. 3.1: A simplified process model in BPMN describing the treatment of patients in a hospital. The branches of the gateway are annotated with transition probabilities based on the case attribute *gender*.

3.1.2 Event Log Data

During the execution of business processes, data is captured and stored in the form of event logs. To formally define the structure of this data, let A be the universe of all process activities, C the universe of case IDs, T the universe of possible timestamps and $S_1, ..., S_m$ the universes of the m attributes associated with the process.

In this thesis, we focus exclusively on **case attributes**, which are static and remain unchanged throughout the execution of a process instance. These attributes describe characteristics of the process instance itself, rather than dynamic properties of individual events. Case attributes can be either **categorical**, containing predefined categories or classes, such as the patients gender or nationality, or **numerical**, representing continuous or discrete numerical values, like the patient's age or income.

In an event log, executed activities are recorded as individual events. Each **event** e can be formally described as a tuple $e = (a, c, s, t)$. Here, $a \in A$ is the activity that has been executed, $c \in C$ is the case ID, linking the event to a specific process instance, s is a tuple $s = (s_1, ..., s_m)$ with $s_i \in S_i, \forall i \in \{1, ..., m\}$, containing the values of each attribute and $t \in T$ is the timestamp denoting when the event occurred.

The sequence of all events $e_1, ..., e_l$ belonging to a single case c, ordered by their timestamps, is called the **trace** of c with length l. Considering that a trace describes the life-cycle of a single case c, all events belonging to a trace have the same attribute values s. A **prefix** with length l' is a portion of such a trace with length l, consisting of the first l' events, with $l' < l$. Finally, an **event log** is a finite collection of traces belonging to the same process.

Case ID	Activity	Timestamp	Gender
1	Register Patient	2025-02-08 09:15:00	Male
1	Expert Treatment	2025-02-08 09:30:00	Male
1	Bill Patient	2025-02-08 09:45:00	Male
2	Register Patient	2025-02-08 10:05:00	Female
2	Regular Treatment	2025-02-08 10:20:00	Female
2	Bill Patient	2025-02-08 10:35:00	Female
3	Register Patient	2025-02-08 11:00:00	Male
3	Expert Treatment	2025-02-08 11:15:00	Male
3	Bill Patient	2025-02-08 11:30:00	Male

Tab. 3.1: Event log for the patient treatment process shown in Figure 3.1, containing only the case attribute

3.1.3 Predictive Business Process Monitoring

Predictive Business Process Monitoring (PBPM) is a field that focuses on analyzing and forecasting the progression of ongoing business processes, based on historical data stored in the form of event logs. This includes tasks such as predicting the remaining time of a case or determining its final outcome. These predictions play a crucial role in enabling organizations to optimize operations, allocate resources effectively, and preempt potential issues. In this thesis, we will take a closer look specifically at **next activity prediction**, where the objective is to forecast the subsequent activity in an ongoing case. Formally, given a prefix $e_1, ..., e_{l'}$ of observed events, the goal is to predict the activity label $a_{l'+1}$ of the next event $e_{l'+1}$.

The task of next activity prediction is categorized as a classification task. These **classification tasks** aim to predict discrete categories, such as determining which event will occur next or whether an outcome is positive or negative. Accordingly, we will refer to models used for next activity as **classifiers**. In contrast, **regression tasks** predict continuous values, such as estimating the remaining time for a process or the monetary amount of a transaction.

3.2 Black Box vs. White Box

Within the field of PBPM, numerous traditional machine learning and deep learning architectures have been employed to develop effective predictive models [RMVL21]. A fundamental distinction among these approaches lies in their categorization as either black box or white box models.

Black box models, such as neural networks, excel in capturing complex patterns and relationships within data but provide limited transparency into their decision-making processes. In contrast, **white box** models, like decision trees or linear regression, prioritize interpretability and simplicity, enabling stakeholders to understand and trust the reasoning behind predictions. However, this may come at the cost of reduced predictive performance for complex tasks. [Rud19]

The difference between black and white box models will become even clearer in the following sections 3.3 and 3.4, as we will look at feedforward neural networks as an example for a black box model and decision trees as an example for a white box model.

3.3 Neural Networks

In this thesis, **feedforward neural networks (NN)** will be used as the main predictor for the task of next activity prediction. While these are not inherently designed for sequence data, such as traces in an event log, their simplicity and ease of training still makes them an attractive choice. Additionallly, the focus of this work doesn't lie on achieving maximum predictive performance, but rather the conceptual demonstration of our approach. Moreover, NNs have been employed in related work [LP24] within PBPM, demonstrating their viability for similar tasks. Nevertheless, since our methodology described in chapter 4 is not specific to using NNs, it is entirely feasible to adapt our approach to other deep learning architectures.

3.3.1 Perceptron

In order to understand NNs, it is helpful to begin with the basic computational unit in a neural network, the perceptron. The perceptron processes an input vector $x = (x_1, ..., x_n)$ in a way that is designed after biological neurons. It combines the input x linearly with a weight vector $w = (w_1, ..., w_n)$ and adds a bias b. The resulting sum is then passed to an activation function σ, which determines the final

output of the perceptron. This activation function typically introduces non-linearity into the computation, such as the ReLU or sigmoid function. [**perceptron**]

Formally, the perceptron calculates its scalar output y for the input x as:

$$y = \sigma\left(\left[\sum_{i=1}^{n} w_i \cdot x_i\right] + b\right) \tag{3.1}$$

3.3.2 Network Architecture

The scalar output of a perceptron can only be used for basic binary classification or regression problems. Even then, due to the simple construction of the perceptron, its effectiveness is highly limited, depending on the complexity of the task. [**perceptron_limited**] NNs extend the perceptron model by organizing multiple perceptrons into fully connected layers, where every perceptron in one layer is connected to every perceptron in the next. This structure allows NNs to handle more complex tasks.

An NN consists of three distinct types of layers, ordered sequentially:

- Input layer: The input layer acts as the interface between raw input data and the network. Each perceptron in this layer corresponds to one feature of the input vector $x = (x_1, ..., x_n)$. The input layer does not perform any transformations, it merely transfers the raw input values to the next layer.

- Hidden layers: Hidden layers are where the network performs the majority of its computations. Each hidden layer contains a set of perceptrons that process data received from the previous layer. These layers are responsible for learning complex representations of the input data, enabling the network to capture intricate patterns and relationships.

- Output layer: The output layer produces the final predictions of the network. For multiclass classification tasks, such as next activity prediction, the layer includes one perceptron for each class, which in our case corresponds to the number of activities.

The number of hidden layers, the number of perceptrons per layer, and the choice of activation functions collectively define the architecture of the network. These design choices are critical in determining the model's ability to handle the task at hand while balancing computational complexity and capacity.

3.3.3 Feedforward Algorithm

With the network architecture defined, the next step is to understand how input data is processed as it passes through the various layers. This process is governed by the **feedforward algorithm**.

The algorithm begins at the input layer, where the input $x = (x_1, ..., x_n)$ is passed directly to the next layer without any transformations. From there, each perceptron in the subsequent hidden layers performs the same operation as provided in formula 3.1, using the outputs from the previous layer as inputs. Formally, the activiation $y_j^{(l)}$ of the j-th perceptron in the l-th layer is computed from the m outputs $y_1^{(l-1)}, ..., y_m^{(l-1)}$ of the previous layer, using weight vector $w^{(l)} = (w_{1,j}^{(l)}, ..., w_{m,j}^{(l)})$, bias $b_j^{(l)}$ and activation function $\sigma_j^{(l)}$ as follows:

$$y_j^{(l)} = \sigma\left(\left[\sum_{i=1}^{m} w_{i,j}^{(l)} \cdot y_i^{(l-1)}\right] + b_j^{(l)}\right) \tag{3.2}$$

A commonly used activation function is the **Rectified Linear Unit (ReLU)** [**relu**], which is defined as $\text{ReLU}(z) = \max(0, z)$.

In the output layer, the perceptrons are activated using the softmax function. Let $z_j^{(l')} = \left[\sum_{i=1}^{m'} w_{i,j}^{(l')} \cdot y_i^{(l'-1)}\right] + b_j^{(l')}$ be the weighted sum of inputs of the j-th perceptron in the output layer L before activation. Then, given the total number of classes k, the softmax activation $y_j^{(l')}$ for the j-th output perceptron is defined as:

$$y_j^{(l')} = \frac{\exp(z_j^{(l')})}{\sum_{i=1}^{k} \exp(z_k^{(l')})} \tag{3.3}$$

The softmax activation ensures that the outputs $y_1^{(l')}, ..., y_k^{(l')}$ form a valid probability distribution, with each value between 0 and 1 and their sum equal to 1. The predicted class corresponds to the one with the highest probability.

All in all, the strictly unidirectional flow of data in the NN ensures that no feedback loops are introduced, maintaining computational simplicity. However, the interplay of numerous weights and biases across multiple layers, combined with the non-linear transformations introduced by activation functions, makes it difficult to intuitively understand or trace how specific inputs lead to specific outputs. This lack of transparency is why we consider NNs to be black box models.

3.3.4 Backpropagation using the Adam Optimizer

To enable the NN to make accurate predictions, its weights and biases must be adjusted based on training data. This process is achieved through the **backpropagation algorithm**, which iteratively updates the network's parameters by minimizing a predefined loss function. In this thesis, we employ the **Adam optimizer** [Kin14], a variant of gradient descent that adapts learning rates per parameter, improving stability and convergence speed.

Training relies on a dataset S composed of **training samples**, where each sample $(x, y) \in S$ consists of an input vector $x = (x_1, ..., x_n)$ with n numerical features and a corresponding target y. For next activity prediction, y is represented as a **one-hot encoded vector** $y = (y_1, ..., y_k)$, where $y_c = 1$ for the target class c and $y_i = 0$ for all other classes $i \neq c$. The total number of classes k corresponds to the number of possible activities.

The discrepancy between the predicted probability distribution $\hat{y} = (\hat{y}_1, ..., \hat{y}_k)$ and the target distribution y is quantified using the **cross-entropy loss** L, which measures how well the predicted probabilities align with the correct class labels:

$$L = -\sum_{i=1}^{k} y_i \log(\hat{y}_i). \tag{3.4}$$

For every sample, training then proceeds in two main steps:

1. **Forward pass:** The input x is propagated through the network using the **feedforward algorithm** (see Section 3.3.3), generating output probabilities $\hat{y}$. The loss function L is then computed based on these predictions and the target labels.

2. **Backward pass:** The gradients of L with respect to the network's weights and biases are computed, starting from the output layer and moving backward through the hidden layers. This process, known as gradient computation, relies on the chain rule of differentiation to propagate errors through the network. The computed gradients indicate how each parameter should be adjusted to reduce the loss.

The parameter updates follow an optimization rule. Instead of standard gradient descent, which updates weights using a fixed learning rate η, we use Adam (**Adaptive Moment Estimation**), which dynamically adjusts learning rates for each parameter by maintaining moving averages of past gradients.

In standard gradient descent, weights are updated as follows:

$$w_{t+1} = w_t - \eta \nabla L(w_t), \tag{3.5}$$

where $\nabla L(w_t)$ is the gradient of the loss function with respect to weight w_t. However, using a fixed learning rate can lead to slow convergence or instability.

Adam addresses this by introducing two modifications:

1. **First-moment estimate (momentum term)**: Instead of using raw gradients, Adam computes an exponentially weighted moving average of past gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L(w_t), \tag{3.6}$$

where β_1 (typically 0.9) controls how much past gradients influence the update. This smooths updates and reduces oscillations.

2. **Second-moment estimate (adaptive scaling)**: To adaptively adjust learning rates, Adam maintains an exponentially weighted moving average of squared gradients:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L(w_t))^2, \tag{3.7}$$

where β_2 (typically 0.999) determines how much past squared gradients contribute. This prevents overly large updates in steep regions of the loss landscape.

Since both estimates are initially biased towards zero, Adam applies bias correction:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{3.8}$$

The final weight update is computed as:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \tag{3.9}$$

where ϵ (e.g., 10^{-8}) ensures numerical stability.

Unlike standard gradient descent, Adam automatically adjusts step sizes for each parameter, leading to faster convergence and improved stability. This makes it particularly effective for deep neural networks.

The parameter updates follow an optimization rule. Instead of standard gradient descent, which updates weights using a fixed learning rate η, we use Adam (**Adaptive Moment Estimation**), which adapts learning rates individually for each parameter by maintaining first and second moment estimates of the gradients. The update rule for a given weight w_t at iteration t is:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L(w_t), \tag{3.10}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L(w_t))^2, \tag{3.11}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{3.12}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{3.13}$$

Here, β_1 and β_2 are decay rates (typically set to 0.9 and 0.999, respectively), and ϵ is a small constant for numerical stability. Unlike traditional gradient descent, Adam dynamically adjusts the step size for each parameter, improving convergence speed and robustness.

Training occurs over multiple **epochs**, where an epoch is a complete pass over the dataset. Instead of computing updates over the entire dataset at once, the data is divided into smaller **batches** of fixed size. Each batch is processed separately, computing gradients and updating weights. Using batches stabilizes training by reducing noise in gradient updates while maintaining computational efficiency. The process repeats for multiple epochs until convergence is reached, defined by a sufficiently low loss or a predefined number of epochs.

By iteratively refining weights and biases through backpropagation and Adam optimization, the NN learns to improve its predictions over time, minimizing classification errors while generalizing well to unseen data.

3.4 Decision Trees

Similarly to NNs, we will also use **Decision Trees** (DT) [Bre17] as a predictor for next activity prediction. One of their most significant strengths lies in their transparency and interpretability, especially when compared to more opaque models like NNs. And despite their lack of an inherent mechanism for handling sequential dependencies, decision trees have been effectively employed in related work [QA19] within the PBPM domain, making them a suitable choice for in this thesis.

3.4.1 Tree Structure and Key Components

In order to understand DTs, we will first go over the structure and key components of a tree in general. [Cor+09] A tree T is a special type of directed graph, defined as $T = (V, E)$, where V is the finite set of **nodes** and $E \subset V \times V$ is the set of directed **edges**. When two nodes $u, v \in V$ are connected by an edged $(u, v) \in E$, we will call u the **parent** of v and likewise v the **child** of u. Similarly, a **path** from u to v, defined a sequence of edges $(u, w_1), (w_1, w_2), ..., (w_{n-1}, w_n), (w_n, v) \subset E$, establishes u as an **ancestor** of v and v as a **descendant** of u.

Within the tree, nodes are classified into either **internal nodes**, when they have at least one child, or **leaf nodes**, when they have no children. Additionally, a node r without parents, formally satisfying $\forall v \in V : (v, r) \notin E$, denotes the **root** of the tree. Accordingly, the **subtree rooted at node** v refers to the tree $T_v = (V_v, E_v)$, where $V_v \subset V$ is the subset containing all descendants of v and $E_v \subset E$ the edges between V_v. In general, $T' = (V', E')$ is a **subtree of** $T = (V, E)$, if $V' \subset V$ and $E' \subset E$. Another essential property of a tree is its **depth**, which measures the longest path from the root node to any of its leaf nodes.

Finally, unlike a general directed graph, a tree must satisfy several strict structural properties:

- **Single Root:** There has to be a unique root node $r \in V$.
- **Connectedness:** Every node $v \in V$ must be a descendant of root node r.
- **Single Parent:** Every node $v \in V$ except the root node r must have exactly one parent $u \in V$.

Furthermore, we will work exclusively with binary trees. These have the additional restriction, that every node must have either exactly two or none children. When a

node has two children, we will assume a fixed distinction into a left child an a right child.

3.4.2 Decision Process

Having presented the necessary definitions, we will now take a closer look at how DTs work internally. In this thesis, input data for DTs will have the same shape as the input for NNs. Therefore an input sample $x = (x_1, ..., x_n)$ is a vector containing n numerical features. When predicting the next activity y for such a sample x, the nodes of the DT serve different roles:

The root node is the starting point for all decision processes. Internal nodes posess a feature index i and a threshold value t. These internal nodes represent decision points, which examine whether the i-th feature x_i of the sample x is greater than the threshold t. Depending on the answer, the sample x is then passed onto either the left or right child. This process continues recursively until the sample reaches a leaf node. Leaf nodes contain a label y, representing the outcomes of the decision process, in our case determining the predicted next activity.

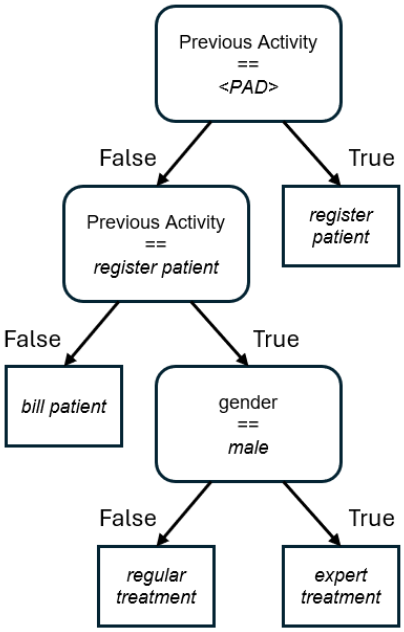

Fig. 3.2: A DT usable for next activity prediction in the process model shown in Figure 3.1. Leaf nodes are depicted as regular rectangles, while internal nodes are shown with rounded corners. Instead of raw feature indices and thresholds, the conditions used for splitting are expressed in natural language for better readability. The previous activity <*PAD*> is a placeholder, if no previous activity has occured yet.

Since the structure of a DT explicitly represents the decision-making process, it becomes immediately clear why they are widely regarded as white box models. Each path from the root to a leaf corresponds to a set of human-readable decision rules,

enabling stakeholders to identify important features and assess the rationale behind each decision.

3.4.3 Training Process

Now that we understand how a DT makes predictions, we turn our focus to the training process—how a DT is built from data. Let S be the collection containing the training data and $|S|$ the amount of elements contained in S. We assume the elements of S to be tuples of the form (x, y), where $x = (x_1, ..., x_n)$ is a sample input and y the corresponding target output label. The training process aims to construct a structure that effectively partitions S in a way, such that the target labels y within the partitions are as homogeneous as possible. This is achieved by selecting the feature and threshold for each split that maximize the "purity" of the resulting subsets.

A common metric used to evaluate this purity in data sets is **Gini impurity**. The Gini impurity measures the likelihood of incorrectly classifying a randomly chosen element, if it were labeled according to the distribution of labels. Intuitively, it quantifies how "mixed" the labels are within S. Mathematically, for k different target labels, where p_i is the proportion of samples in S having the i-th target label, the Gini impurity is defined as:

$$G(S) = 1 - \sum_{i=0}^{k} p_i^2 \tag{3.14}$$

The Gini impurity $G(S)$ ranges from 0 to $1 - \frac{1}{k}$. A value of $G(S) = 0$ indicates that S is perfectly pure, with all samples having a single target label, while $G(S) = 1 - \frac{1}{k}$ represents a maximally impure dataset where samples are evenly distributed among all k target labels. Generally, a lower $G(S)$ implies a higher purity. [Bre17]

The training process begins at the root node, which is associated with the entire training dataset S. At each step, the algorithm evaluates all possible splits of the data by testing every feature x_i and various thresholds. Thresholds for splitting are derived directly from the values in the dataset associated with the node, ensuring that every threshold capable of separating samples into distinct subsets is tested. Specifically, given a feature x_i and a sorted list of its values in the associated dataset, potential thresholds are chosen as the midpoints between consecutive unique values. Splits are chosen greedily, selecting the feature and threshold that provide the largest reduction in impurity. Formally, if splitting S into S_1 and S_2 results in Gini

impurities $G(S_1)$ and $G(S_2)$, the split is selected to minimize the weighted Gini impurity G_{split}:

$$G_{split}(S_1, S_2) = \frac{|S_1|}{|S|} G(S_1) + \frac{|S_2|}{|S|} G(S_2) \tag{3.15}$$

Once the best split is determined, two child nodes are created, and each is associated with one of the datasets S_1 and S_2 resulting from the split. This process is recursively repeated for each child node until a stopping criterion is met, such as reaching a maximum tree depth, achieving a minimum number of samples per leaf, or reducing impurity to a negligible level.

At the end of this recursive process, the leaf nodes of the tree represent the terminal points of the decision-making structure. Each leaf is associated with a subset of the training data that corresponds to the sequence of splits leading to that leaf. For our task of next activity prediction, the output at each leaf is determined through majority voting: the output label y that occurs most frequently in the training samples associated with the leaf is selected as the predicted label.

3.4.4 Cost Complexity Pruning

Depending on the selected stopping criterion, the final DT may include subtrees with numerous nodes that contribute little to the tree's predictive value. This not only makes it more difficult to examine the tree's structures, but may also hurt the overall accuracy of the DT due to overfitting. To address this, pruning is a common technique used to simplify decision trees by removing unnecessary branches, ultimately improving the tree's performance and interpretability.

One effective approach to pruning is **minimal cost-complexity pruning** [Bre17]. This algorithm utilizes a complexity parameter $\alpha \geq 0$, which balances the trade-off between a tree's complexity and its predictive performance. Based on α, the cost-complexity measure C_α of a given tree T is defined as

$$C_\alpha(T) = R(T) + \alpha|T|, \tag{3.16}$$

where $|T|$ is the number of leaf nodes in the tree, and $R(T)$ represents the total impurity of T, computed as the total sample-weighted Gini impurity of the leaf nodes in T. Minimal cost-complexity pruning finds the subtree $\bar{T}$ of T, such that the cost-complexity measure $C_\alpha(\bar{T})$ is minimized. Since α penalizes tree complexity, larger values lead to more aggressive pruning.

The algorithm works by iteratively removing nodes from T. Every iteration starts with computing the effective cost-complexity value α_t for each internal node t as

$$\alpha_t = \frac{R(t) - R(T_t)}{|T_t| - 1}, \tag{3.17}$$

where T_t is the subtree rooted at t.

The inner node t with the smallest α_t is refered to as the weakest link in the tree. If $\alpha_t \leq \alpha$, this node and its subtree are pruned, meaning all its child nodes are removed, and it becomes a leaf node. The pruning process continues iteratively until the minimal α_t in the remaining tree exceeds the predefined pruning parameter α, ensuring that the final tree retains only splits that contribute meaningfully to impurity reduction.

3.5 Evaluation Metrics

Having introduced both NNs and DTs, we will go over the metrics we need to evaluate the performance of these models in our experiments.

3.5.1 Accuracy

Accuracy is a fundamental performance metric in machine learning that measures the proportion of correctly predicted instances relative to the total number of instances. For next activity prediction in PBPM, accuracy evaluates how well the model correctly predicts the upcoming event in a process trace. Given a set of predictions, the accuracy of the model is defined as:

$$\text{Accuracy} = \frac{\#(\text{Correct predictions})}{\#(\text{Total predictions})} \tag{3.18}$$

3.5.2 Demographic Parity

While accuracy is a crucial metric for assessing overall model performance, it does not account for potential biases in decision-making. This necessitates the inclusion of fairness-aware quantitative measures for assessing and comparing models to ensure that they meet fairness criteria alongside performance metrics.

One possible perspective on discrimination is provided by **group fairness** metrics, which offer a way to assess whether a machine learning model's predictions are

equitable across different groups within the population. In PBPM, these metrics help ensure that predictions, such as the next activity or outcome in a business process, do not favor one group over another based on sensitive attributes. These **sensitive attributes** refer to characteristics of individuals or groups that are typically protected by laws or ethical guidelines due to their potential to lead to biased or discriminatory treatment, such as gender, ethnicity, age, disability status, etc.

In this thesis, we utilize on one of the most prevalent fairness metrics, **demographic parity** [Dwo+12], which quantifies the disparity in positive prediction rates across different demographic groups. Let y be the prediction of the model, that is either categorized as positive for $y = 1$ or negative for $y = 0$. Moreover let S be a sensitive attribute whose values can be split into the demographic groups s_1 and s_2. A model satisfies demographic parity if the probability of receiving a positive prediction $y = 1$ is independent of the sensitive attribute S, meaning:

$$P(y = 1 \mid S = s_1) = P(y = 1 \mid S = s_2) \tag{3.19}$$

Based on the definition of demographic parity, the discrimination can be measured as the difference in selection rates, commonly referred to as ΔDP, similar to how it is defined in [QA19]:

$$\Delta DP = |P(y = 1 \mid S = s_1) - P(y = 1 \mid S = s_2)| \tag{3.20}$$

Since the true probabilities are unknown, they are estimated from the model's predictions for available samples, using the observed selection rates for each demographic group as:

$$P(y = 1 \mid S = s_i) \approx \frac{\#(\text{Samples} \mid y = 1, S = s_i)}{\#(\text{Samples} \mid S = s_i)}, i \in \{1, 2\} \tag{3.21}$$

ΔDP ranges between the values 0 and 1. A smaller ΔDP indicates that the model's predictions are more equitable across demographic groups, whereas a larger ΔDP suggests a potential bias in the model's decision-making process.

As an example, let's consider the process model shown in Figure 3.1, where a gateway after the *register patient* activity determines whether a patient is assigned to an *expert examination* or a *regular examination*. To evaluate whether a machine learning model makes fair predictions at this decision point, we can use samples where the previous activity was *register patient*. These samples are split into demographic groups based on *gender*, distinguishing between *male* and *female* patients. In this context, the positive prediction is defined as the assignment to *expert examination*, while any other outcome is considered negative.

Suppose the model assigns 70% of male patients to *expert examination* and 30% of female patients to *expert examination*, as is done in the underlying process model. The ΔDP is then computed as $\Delta DP = |P(y = \textit{expert examination} \mid \textit{gender} = \textit{male}) - P(y = \textit{expert examination} \mid \textit{gender} = \textit{female})| = |0.7 - 0.3| = 0.4$.

This result reflects the disparity in selection rates, indicating that the model disproportionately assigns one of the demographic groups to expert examinations. By measuring and minimizing ΔDP, we aim to ensure that predictive models used in PBPM do not systematically favor one demographic group over another, thus improving fairness in the decision-making processes.

3.6 Formalized Problem Statement

Having cleared all the preliminaries, we can now present a formalized problem statement. Given a classifier C_{enriched} for next activity prediction, which has access to all sensitive attributes, its predictions may inadvertently exhibit unfairness for certain decisions and demographic groups, as measured by demographic parity disparity (ΔDP). The objective of this thesis is to mitigate such unfairness while maintaining high predictive performance.

To address this issue, we aim to derive a modified classifier C_{modified} that reduces the observed ΔDP in C_{enriched}, thereby enhancing fairness in decision-making. However, this fairness enhancement should not come at the cost of excessive performance degradation. Therefore, C_{modified} must achieve an accuracy greater than that of a baseline classifier C_{base}, which does not utilize any sensitive attributes and is, by design, demographically fair.

The reduction of unfair demographic disparities is guided by domain-specific considerations, ensuring that only those disparities deemed unfair within the context of the process are addressed. This is achieved through a knowledge distillation approach, which enables a domain expert to analyze and refine the decision-making process of the model. The precise details of this methodology are outlined in the next chapter.

4 Methodology

This chapter will outline the complete pipeline of our methodology. It describes the process of data gathering and preprocessing, with a focus on handling bias causing attributes. The chapter then explains how the initial ML model is trained, followed by the application of knowledge distillation to create an interpretable representation of the model, which can then be modified to address unwanted biases. The methodology further includes how the predictions of the modified representation can be used to finetune the original model in order to improve fairness, while maintaining accuracy.

4.1 Data Gathering

Public real life event logs containing sensitive attributes are hard to come by. Additionally, in order to properly highlight the full capacity of our method, we require the data to have sensitive attributes that cause both positive and negative bias respectively. Although [Poh+23] recently published simulated event logs, seeking to address the scarcity of fairness-aware datasets in PBPM, those event logs didn't show the structure we were looking for in this thesis. Therefore, we will create the necessary data ourselves, either by simulating a process or by enriching a real life event log with sensitive attributes.

4.1.1 Event Log Simulation

Our simulated event logs are generated by following rule-based transitions within a constructed process model. Each case starts at an initial activity and progresses through sequential transitions to the next activity, following the predefined sequence flow until reaching the end. When reaching a gateway, the transition probabilities are determined by the values of the case attributes. These values are being generated randomly for each process instance. Additionally, for each transition, there is a 1% chance that the process will deviate to a randomly selected activity, even if there is no connecting sequence flow to that activity This introduces variability and some realism into the simulation by occasionally branching out from the standard sequence of activities. The timestamps in the simulated data do not convey meaningful

information. Instead, the time intervals between activities are randomly assigned between one and ten minutes.

An example of a simple process model, describing the treatment of patients in a hospital, is shown in Figure 4.1. This model consists of only four activities and uses *gender* as its only case attribute. The attribute *gender* can take one of two values, *male* or *female*, each with an equal probability of 50%. In this example, *gender* determines the probabilities at the gateway following the activity *register patient*. For *male* patients, the probability of transitioning to *regular treatment* is 70%, while the probability of transitioning to *expert treatment* is 30%. Conversely, for *female* patients, the probability of transitioning to *regular treatment* is 30%, and the probability of transitioning to *expert treatment* is 70%.

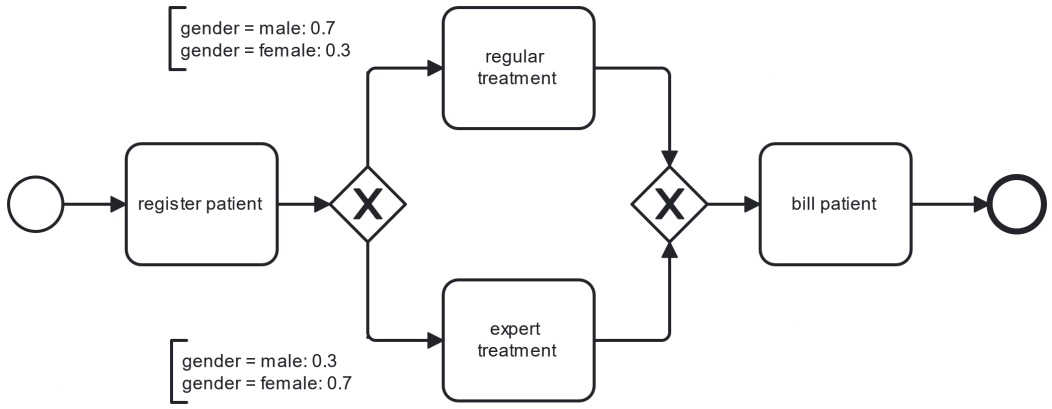

Fig. 4.1: A simplified process model in BPMN describing the treatment of patients in a hospital, as portrayed before in Figure 3.1. The branches of the gateway are annotated with transition probabilities based on the case attribute *gender*. We will use it as an example again in this chapter.

4.1.2 Event Log Enrichment

A common critique of using simulated event logs is that their complexity may not fully reflect that of event logs derived from real-life processes. To address this, we also augment publicly available event logs, which lack sensitive attributes, by adding these attributes ourselves. This enrichment process follows a set of predefined probabilistic rules, that depend on the sequence of events observed in a case. The case attributes are then assigned according to the distributions that aligns with the first matching rule. Detailed examples of these enrichment rules and their corresponding attribute distributions are provided in chapter 5.

4.2 Preprocessing

Since NNs cannot learn from the event log format directly, we preprocess the data to generate samples in the form of (x, y), where $x = (x_1, ..., x_n)$ represents the n input features and $y = (y_1, ..., y_k)$ represents the target outcome out of k activities, as described in section **??**. In order to do this, we extract all possible prefix-outcome pairs from each case in the event log. The prefix consists of the sequence of events up to a certain point in the case, while the outcome corresponds to the next activity after the prefix. Our target y can be directly derived from the next activities by one-hot encoding.

The input x should contain information about both the prefix and the corresponding case attributes. Since neural networks require inputs of a fixed size, we handle the varying lengths of prefixes by applying a sliding window of a static size w. If a prefix is shorter than the window size, we pad the sequence with a special <*PAD*> activity to fill the remaining positions. Conversely, if a prefix is longer than w, we truncate it to include only the most recent w events.

Case attributes are encoded based on their type. Categorical attributes are one-hot encoded in the same way as activities, ensuring that each category is represented as a unique vector. For consistency purposes, we will also one-hot encode categorical attributes that have only two possible values, even though a single bit would suffice for Numerical attributes are normalized using **min-max scaling**, which rescales their values to fall within the range of 0 to 1, ensuring that all attributes contribute equally to the model's training process, preventing numerical attributes with larger magnitudes from dominating those with smaller ones, as well as categorical attributes. Given a numerical value z, the scaled value z' is calculated as

$$z' = \frac{z - min}{max - min} \tag{4.1}$$

where *min* and *max* refer to the minimum and maximum observed values of the attribute in the dataset. Finally, we concatenate the sequence of one-hot encoded activities and the encoded case attributes into the input vector x.

After processing all samples, we have to consider how to split the samples into a **training set** and a **test set**. The training set is used to improve the NN's accuracy, while the test set is reserved for the final evaluation of the model's performance after training is complete. It is crucial to keep these sets strictly separated to prevent any information from the test set leaking into the training set. Therefore, special care must be taken during the splitting process: [WDW21]

Previous Activity	Encoding
<PAD>	[1, 0, 0, 0, 0]
register patient	[0, 1, 0, 0, 0]
regular treatment	[0, 0, 1, 0, 0]
expert treatment	[0, 0, 0, 1, 0]
bill patient	[0, 0, 0, 0, 1]

Gender	Encoding
male	[1, 0]
female	[0, 1]

Previous Activity	Gender	Next Activity	x Encoding	y Encoding
register patient	male	expert treatment	[0, 1, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0]
expert treatment	male	bill patient	[0, 0, 0, 1, 0, 1, 0]	[0, 0, 0, 0, 1]
register patient	female	regular treatment	[0, 1, 0, 0, 0, 0, 1]	[0, 0, 1, 0, 0]

Tab. 4.1: Tables portraying the one-hot encoding of the previous activities (top left) and genders (top right), as well as the input-output mappings for some samples from the model presented in Figure 4.1 (bottom).

- **Min-Max Scaling:** When performing min-max scaling, the scaling parameters min and max are derived from just the training set. This simulates a real-world scenario in which only the values of old training data is known to us, which ensures that the test set remains independent.

- **Instance-Level Leakage:** We ensure that all events of a single case are contained either in the training set or the test set, but not both. Splitting process instances across sets might lead to information leakage, allowing the model to gain knowledge about test data it would not have access to in a real-world scenario.

- **Temporal Leakage:** When utilizing timestamp date, one could consider using a chronological split to train on earlier cases and test on later ones. This setup mimics real-world predictive scenarios where historical data is used to predict outcomes for future events. In our case, we mostly use simulated event logs, in which there is no chronological difference, or public event logs, in which the timestamps have been anonymized, maintaining only the time difference between events. [Man17] Therefore, a random split can be used instead.

- **Cluster Leakage:** If the dataset contains groups of highly similar cases, e.g., cases grouped by customer, department, or product line, assigning entire clusters to either the training or test set prevents correlated information from leaking between the splits. The datasets we use however are either simulated or anonymized to protect personal information [Man17]. As a result, no distinct clusters are evident, making a random split both appropriate and sufficient.

- **Representative Datasets:** Lastly, it's important to ensure that the test set is representative of the dataset's overall diversity. Specifically, one should avoid creating splits where rare or common cases are disproportionately represented in one set, as this could skew model evaluation and lead to biased results. Since we are working with rather large datasets, any bias introduced by random splitting is likely to be minimal. Moreover, our primary objective is not to achieve maximum accuracy but to enable a fair and consistent comparison between models, making randomly splitting acceptable for this task.

4.3 Training the Neural Network

After processing the event log into a dataset suitable for machine learning, we proceed to train a feedforward neural network (NN) using the prepared samples. The model is implemented using the the Keras [Cho15] library, which allows for a straightforward layer-by-layer construction of the network. As mentioned earlier, our primary focus is not on fully optimizing accuracy, so we did not invest extensive time or effort into selecting and fine-tuning the model's hyperparameters. The following architecture and hyperparameters were chosen, because they delivered consistently good results in practice:

The NN incorporates four hidden layers with 512, 256, 128, and 64 neurons The choice of progressively smaller layer sizes allows the network to gradually distill the input features into more abstract representations, facilitating the learning of hierarchical patterns. The amount of perceptrons in the input and output layer is dependent on the task at hand, determined by the dimensions of the processed input and output samples respectively. The hidden layer uses the ReLU activation function, while the ouput layer uses softmax activation, as described in section 3.3.3.

For the training process, we used categorical cross-entropy loss in conjunction with the Adam optimizer, utilizing its default parameters as provided by Keras, with a learning rate of $\alpha = 0.001$. The model is trained for a total of 10 epochs, with a batch size of 32. In our experiments, 10 epochs were sufficient for the model to converge without overfitting. The batch size of 32 was chosen because it offers a practical compromise between computational efficiency and stable gradient updates. This configuration enabled the model to learn effectively and produce satisfactory results within a reasonable training time.

4.4 Knowledge Distillation

After training, the NN is expected to achieve high accuracy but may produce biased predictions, hidden by its black-box nature. To uncover these biases and better understand the NN's inner workings, we train a transparent model that mimics the NN's behavior.

This approach leverages **knowledge distillation**, a technique originally developed to transfer knowledge from a large, complex model **(teacher)** to a smaller, more efficient model **(student)** [HVD15]. By training the student to replicate the teacher's outputs instead of the original training targets, knowledge distillation enables faster inference and reduced computational demands, while preserving predictive performance. In addition to efficiency, knowledge distillation has been adapted for interpretability by training a white-box student model to approximate the teacher model's predictions. [LWM18] This allows the decision-making process of the opaque NN to be examined through the interpretable structure of the student, providing insights into its behavior while retaining much of its predictive accuracy.

We opted to use regular decision trees (DTs) as defined in section 3.4, since they are both easy to implement and can be interpreted well even by domain experts not too familiar in the field of machine learning. However, DTs cannot directly utilize the softened output $\hat{y}$ produced by the softmax activation in the output layer of the neural network, as they require discrete class labels rather than a probability distribution. To address this, we apply the *argmax* function to select the class index with the highest probability from the distribution. Formally, given a training sample x, the distilled target labels $\bar{y}$ are derived from the NN's output as follows:

$$\bar{y} = argmax(NN(x)) \tag{4.2}$$

4.5 Training the Decision Tree

Using the input samples x from the training dataset and their corresponding distilled target samples $\bar{y}$ from the NN, we proceed to train our DT. For this, we employ the **DecisionTreeClassifier** class from the Sklearn library [Ped+11], utilzing its default parameters for training. As outlined in section 3.4.3, the DT is constructed by recursively partitioning the training dataset at each node to minimize Gini impurity, until either all leaves contain pure target classes or until further splits no longer reduce impurity.

Additionally, minimal cost-complexity pruning is applied with $\alpha = 0.001$, eliminating splits that provide negligible reductions in impurity. Selecting an appropriate α is important, as excessivly larger values may remove important nodes that rely on sensitive attributes, making it impossible to modify the tree structure as intended. A more robust approach would involve computing the cost-complexity pruning path by iteratively increasing α and recording the sequence of pruned subtrees. This process generates a series of increasingly simplified trees, each corresponding to a different value of α. The optimal subtree is then selected based on a validation criterion, such as minimizing prediction error. However, due to computational constraints and the fact that the results were satisfactory without fine-tuning α, we opted for a fixed value instead.

4.6 Modification of the Decision Tree

Now that the inner workings of the NN have been represented as a decision tree, we can examine its structure for potential biases. Specifically, we look for inner nodes that use sensitive attributes as the basis for their splits and assess whether such usage is justified in the context of the node. If we determine that one or more nodes unfairly rely on sensitive attributes, we can modify the decision tree structure by deleting those biased nodes, thereby removing their influence. To remove a biased node v while maintaining the defined structure of a binary DT, we have several possible options:

- **Cutting Branches:** Let u_1, u_2 be the children of v and $|S_{u_1}|, |S_{u_2}|$ the number of training samples associated with each child respectively. When removing the node v from the tree, the simplest approach would be to replace v with the child node u_i that has more associated training sampels $|S_{u_i}|$, with $i \in \{1, 2\}$. Basically, we would eliminate the subtree rooted at the child with less samples. This method ensures that the majority of samples are still handled similarly, while removing the biased split from the DT. However, the subtree and its splits may lose their coherence without the biased split, potentially leading to a significant drop in predictive performance after the modification.

- **Retraining Subtrees:** An alternative approach when removing v is to delete the entire subtree rooted at v and retrain it using the samples associated with v. The retraining process follows the procedure outlined in section 3.4.3, stopping when the dataset becomes fully pure or when the retrained subtree reaches a greater depth than the subtree rooted at v had before. To prevent the retrained subtree from making the same biased split however, any features related to the sensitive attributes used by v must be excluded from the entire subtree.

Although this results in more coherent splits, there is a possibility that the sensitive attribute could have been used by a descendant of v in a beneficial way, which we might not want to eliminate. Additionally, the retraining process could introduce new biases related to other sensitive attributes that were not previously present. This means the retraining may need to be repeated multiple times until all unwanted bias is removed.

- **Manual Modification:** Finally, a domain expert could manually modify the decision tree, its node structure, and the associated features and thresholds based on their understanding of how the splits should be structured. However, even with domain knowledge, manually adjusting the decision tree requires significant effort and is challenging to do without substantially compromising prediction accuracy.

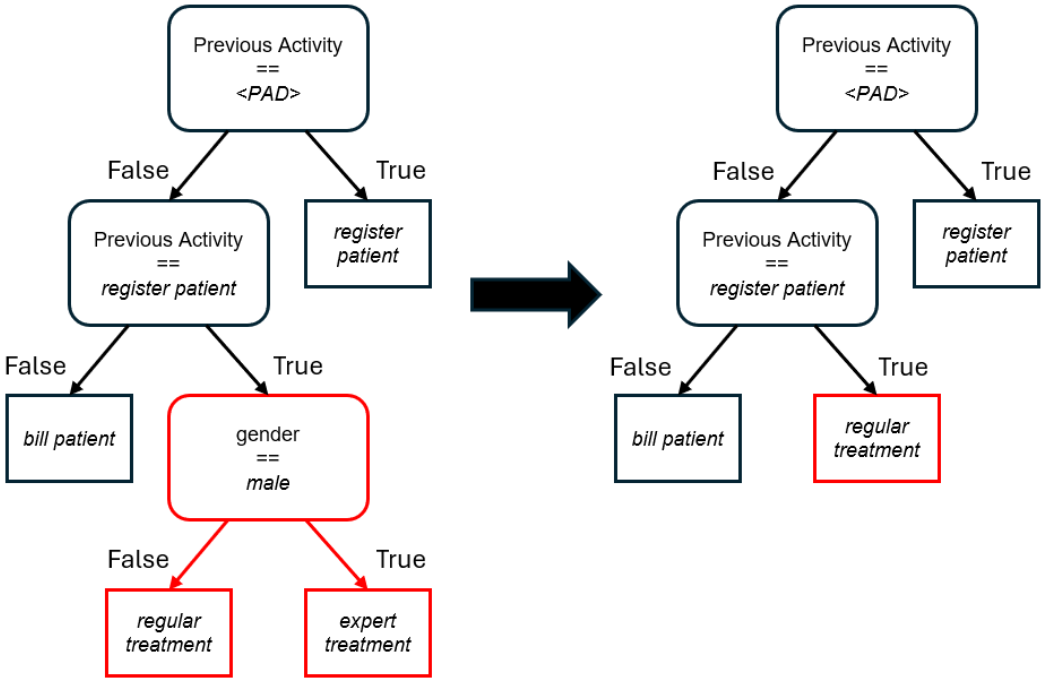

Fig. 4.2: Two DTs suitable for next activity prediction in the process model shown in Figure 4.1. The DT on the left represents the original model, which uses gender discriminatingly in the subtree highlighted in red. Since no alternative attribute is available for making this decision, one possible modification to improve fairness would be to remove the branch leading to *expert treatment*, resulting in the modified DT on the right.

4.7 Finetuning of the Model

By modifying the DT, our goal was to obtain a predictor that makes fairer decisions, while maintaining most of the NN's predictive power. While we could just use the DT directly for the task of next activity prediction, deep learning methods generally

outperform traditional machine learning approaches, when it comes to solving complex tasks [Kra+21]. In order to combine the more precise predictions of the NN with the fairer decisions of the DT, we attempt to fine-tune our NN in accordance to the predictions of the modified DT.

Generally speaking, **fine-tuning** [PY09] is a transfer learning technique that involves adapting a pre-trained model to a new task or dataset. Rather than training a neural network from scratch, fine-tuning starts with a model that has already been trained on a related task, leveraging its pre-learned features. This reduces training time, requires fewer data, and often improves performance by building on the general representations learned during pre-training.

In our case, we fine-tune the pre-trained NN from the previous section 4.3, using a training set that was modified to increase group fairness. This fine-tuning dataset retains the same input samples x from the original training set, but leverages the outputs from the modified DT to replace the original targets y. However, we cannot utilize the immediate output of the modified DT directly, since NNs require distributions as their targets, instead of the discrete labels provided by the DT. To address this, we apply one-hot encoding to the output of the modified DT, receiving a vector $\tilde{y}$ for a training sample x as follows:

$$\tilde{y} = \textit{one-hot}(DT_{mod}(x)) \tag{4.3}$$

When selecting appropriate targets for the training set during fine-tuning, we explored several strategies. The following two approaches yielded the best results in our experiments:

- **Full Correction**: In the first approach, the NN is trained to directly mimic the modified DT by using $\tilde{y}$ as the new training target for all samples. This ensures that the NN fully inherits the fairness properties of the DT, but also means that any misclassifications made by the modified DT are learned as well, even if those are independent of the fairness concerns. As a result, the accuracy of the NN may decrease further than necessary.

- **Selective Correction**: An alternative approach selectively utilizes $\tilde{y}$ as targets only for samples where the modified DT's predictions differ from those of the original distilled DT. Otherwise, we retain the original NN's predictions as training targets, such that the learned parameters are being changed as little as possible. This method ensures that the NN specifically corrects unfair classifications that, thereby mitigating the risk of reduced accuracy caused by erroneous DT predictions.

Although **selective correction** is usually the better choice, it doesn't perform better for all event logs we tested. Therefore, when finetuning, we employ both described approaches and pick the one that yields the better accuracy results. The training process during fine-tuning was similar to the initial training procedure described in section 4.3, with minor adjustments. We did not conduct extensive tests regarding the fine-tuning hyperparameters, since the ideal settings are heavily dependent on both the dataset and the architecture of the black-box predictor. However, these selected hyperparameters yielded satisfactory results in our experiments: We reduced the Adam optimizer's learning rate α from 0.001 to 0.00001 to balance fairness improvements with maintaining the pre-trained weights. Additionally, the number of epochs was reduced from 10 to 5, as the NN's accuracy converges quicker on the modified distilled targets.

5 Evaluation

In this chapter, we systematically assess the performance and key characteristics of the proposed methodology. We begin by outlining the evaluation metrics and experimental framework before presenting the results across multiple datasets.

5.1 Experimental Framework

As outlined in the formalized problem statement, we employ three distinct model configurations for evaluation:

- **Base Model:** The base model operates without access to any sensitive attributes and serves as a baseline for comparison with the other models. Its inputs are limited to recent activities, the time delta since the last event and attributes that are considered to be non-sensitive.

- **Enriched Model:** The enriched model leverages all available sensitive attributes to maximize its performance, fully exploiting the dataset's potential. However, this approach prioritizes accuracy at the expense of fairness, representing the biased model that we aim to improve using our methodology.

- **Modified Model:** The resulting model, when applying our proposed fairness-enhancing methodology to the enriched model.

When we employ our methodology during evaluation, we want to avoid having to alter the distilled decision trees by hand. Therefore, we automatically remove any node from the distilled decision tree that uses a sensitive attribute as its feature for splitting, if the node's subtree contains leaves where the sensitive attribute introduces negative bias in the output classes. When removing such nodes, we use the method of *retraining subtrees*. Since this method may introduce new unwanted biases, we repeat the process of finding and removing unwanted subtrees as often as there are sensitive attributes. In subtrees, where individual features have been removed for retraining, these features stay removed over all iterations of *retraining*.

To ensure robust evaluation of these models, we utilize **5-fold cross-validation** [Koh95]. This method involves partitioning the dataset into five approximately equal subsets called **folds**. In five iterations, each fold serves as a test set exactly once, while the remaining four folds are used for training the model. We evaluate each model in each iteration using accuracy as the metric for predictive performance and demographic parity to measure the effect of negative bias from sensitive attributes.

5.2 Cancer Screening

Our first dataset was created by simulating the cancer screening process in a hospital. The underlying process model, consisting of ten distinct activities, can be seen in Figure 5.2. The dataset includes 10,000 simulated cases, each with the numerical attribute *age* and the categorical attribute *gender*. While we consider *gender* to be a sensitive attribute, *age* is a non-sensitive attribute in this process. *Age* follows a normal distribution with a mean of 45 and a standard deviation of 10, constrained between 20 and 85, while *gender* is assumed to be binary, evenly distributed at 50% each.

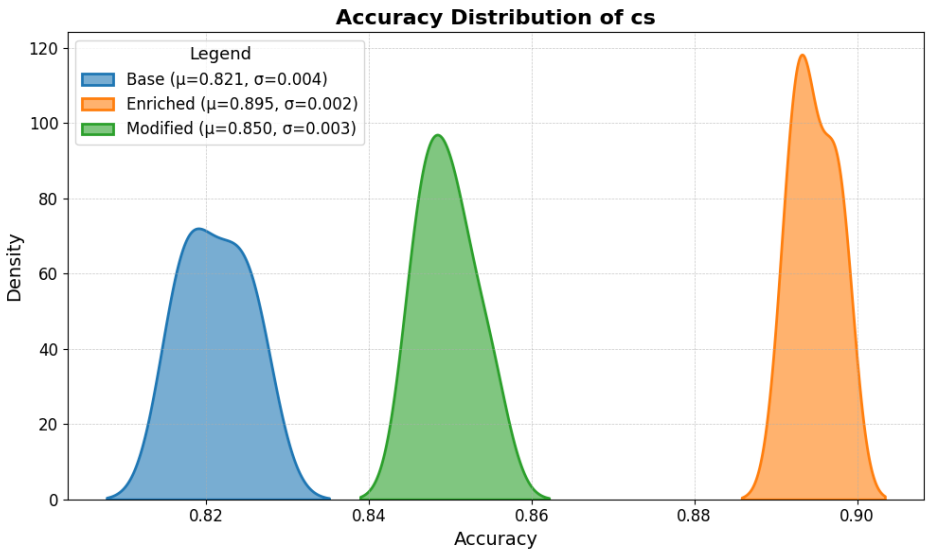

Fig. 5.1: Distribution of accuracy values for the base, enriched, and modified models on the *cancer screening (cs)* event log. The plot illustrates kernel density estimates of accuracy, with shaded regions representing the distribution spread.

Age influences whether issues are detected during screening, as patients over the age of 60 have a 70% probability of proceeding to *explain diagnosis* and a 30% probability of *inform prevention*, whereas those probabilities are reversed for patients aged 60 or younger. *Gender* directly determines whether a patient undergoes *prostate screening* or *mammary screening*, with probabilities of 100% or 0% respectively. Since these distinctions are medically justified, no fairness correction is necessary so far.

However, *gender* also introduces an unfair bias when assessing eligibility, where male patients have a 70% likelihood of being accepted into *collect history* and a 30% likelihood of being sent to *refuse screening*, while for female patients, these probabilities are reversed. Therefore, when evaluating the fairness of our models, we calculate ΔDP for the positive outcome *collect history* between male and female patients for samples with the previous activity *assess eligibility*.

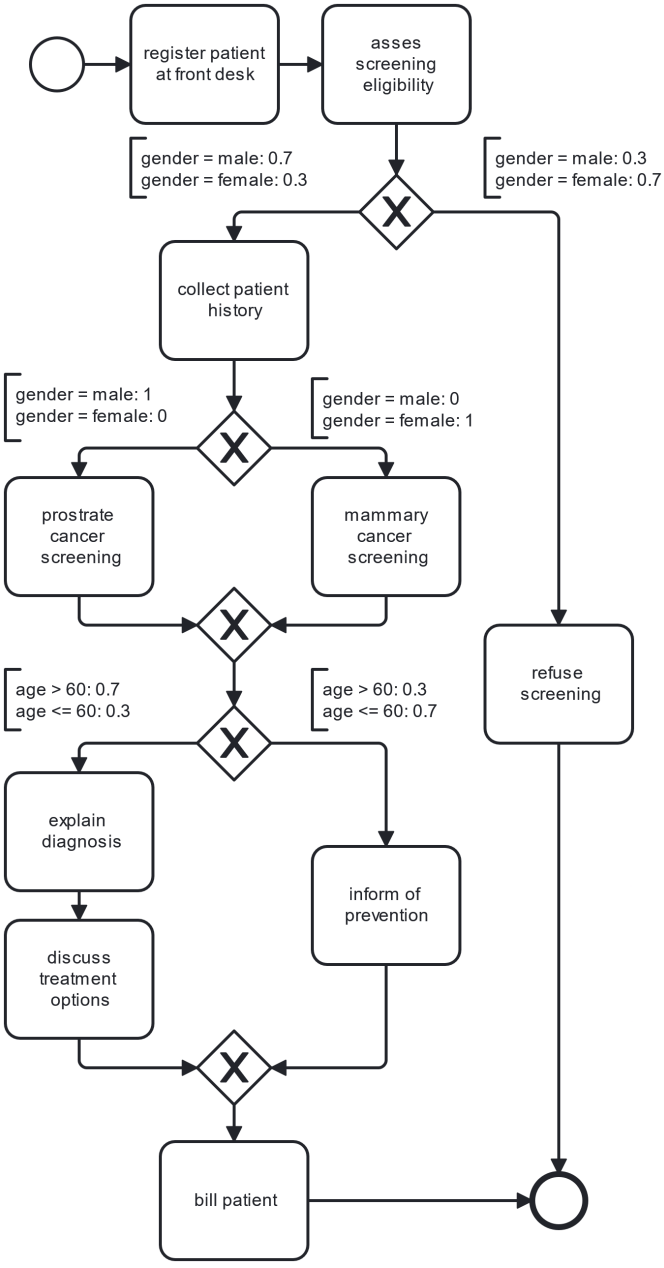

Fig. 5.2: This figure shows the BPMN diagram of the process model used for simulating the *cancer screening* event log. The branches of the gateways are annotated with the underlying conditions based on the case attribute values and their corresponding probabilities.

5.3 Hospital Billing

The **Hospital Billing** [Man17] event log is a widely recognized dataset in process mining and healthcare analytics. Extracted from the financial modules of a hospital's ERP system, this event log records the sequence of activities involved in billing medical services. The dataset comprises 100,000 cases, representing a randomly selected sample of process instances recorded over three years. It includes a total of 17 distinct activities, capturing various steps in the billing workflow.

To optimize computational efficiency for both training and testing, we follow the approach in [QA19] and use only the last 20,000 cases. It is important to note that events and attribute values have been anonymized to ensure confidentiality. For this purpose, the timestamps have been randomized as well, with only the relative time intervals between events within a case preserved. As a result, the last 20,000 cases do not necessarily correspond to the most recent process instances.

Unfortunately, the event log doesn't contain any case attributes we can use for our experiments. Therefore, we will enrich the data with the numerical case attribute *age* and the categorical attribute *gender*. For this event log, we consider both of these attributes to be sensitive. A preliminary analysis of the **Directly-Follows Graph (DFG)**, a process model representation that captures transitions between activities, reveals relatively few decision points where bias could be introduced. However, two key points in the process stand out:

- Whether a case includes the activity *CHANGE DIAGN*, indicating a modification in the assigned diagnosis.

- Whether, after the *RELEASE* activity, the case transitions to *CODE OK* or *CODE NOK*, where *CODE OK* signifies successful billing, while *CODE NOK* indicates a rejected or problematic case.

Since changes in diagnosis may be medically justified, we do not consider biases related to *CHANGE DIAGN* as problematic. However, demographic disparities in the assignment of *CODE OK* and *CODE NOK* raise fairness concerns. Given the relatively small number of cases that transition to *CODE NOK*, the artificially embedded biases in age and gender must be strong enough to create a measurable disparity in the enriched model.

We create four experimental conditions by considering all possible combinations of bias and no bias for the enriched attributes *age* and *gender*. This setup enables us to evaluate how each biased attribute individually, as well as in combination,

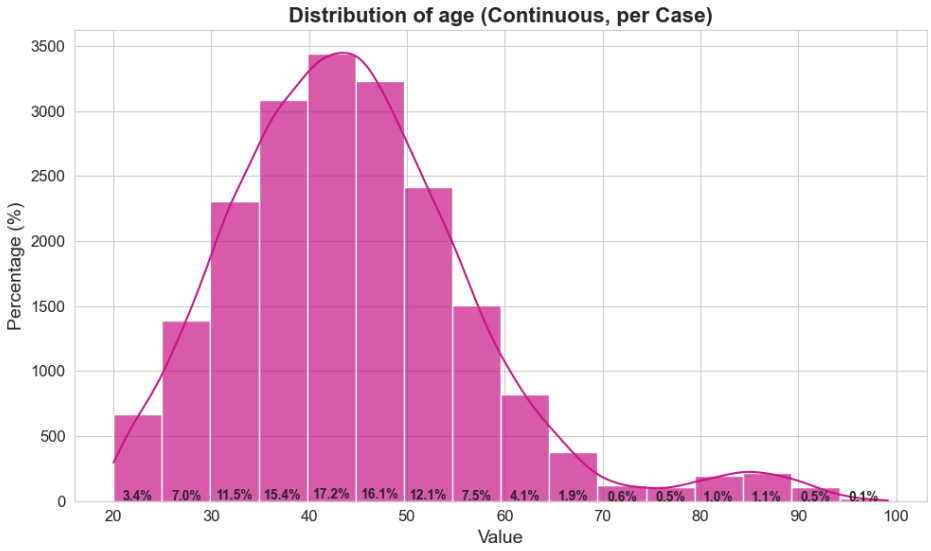

Fig. 5.3: The distribution of the attribute *age*, when it causes bias in the *hospital billing* event log.

influences the fairness of the process. If *age* does not introduce bias, it follows a normal distribution with a mean of 50, a standard deviation of 15, and is constrained between 20 and 100. If *gender* does not introduce bias, it is distributed as 49.5% *male*, 49.5% *female*, and 1% *non-conforming*. If *age* causes bias, the assignment follows these probabilistic rules:

- **If** the event sequence contains *CODE NOK*, **then** age follows a normal distribution with mean 85, standard deviation 5, and is limited between 20 and 100.

- **Else if** the event sequence contains *CHANGE DIAGN*, **then** age follows a normal distribution with mean 50, standard deviation 10, and is limited between 20 and 85.

- **Else** (if none of the above conditions apply), **then** age follows a normal distribution with mean 40, standard deviation 10, and is limited between 20 and 85.

The resulting distribution is visualized in Figure 5.3. And if *gender* causes bias, the probabilistic rules are:

- **If** the event sequence contains *CODE NOK*, **then** *male*: 10%, *female*: 10%, *non conforming*: 80%.

- **Else if** the event sequence contains *CHANGE DIAGN*, **then** *male*: 29.5%, *female*: 69.5%, *non conforming*: 1%.

- **Else** (if none of the above conditions apply), **then** *male*: 69.95%, *female*: 29.95%, *non conforming*: 0.1%.

These rules result in an assigment of around 55.8% male, 41.4% female and 2.8% non-conforming patients.

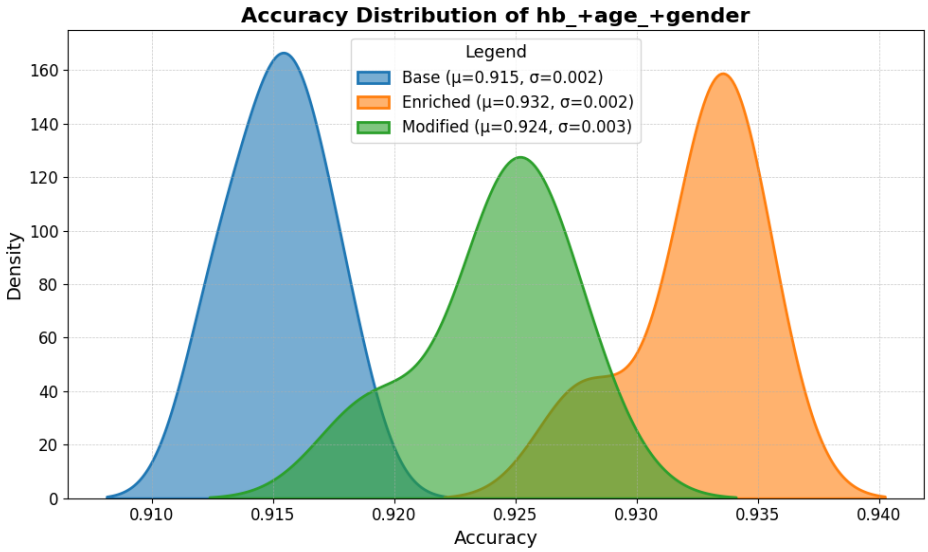

Fig. 5.4: Distribution of accuracy values for the base, enriched, and modified models on the *hospital billing* event log, where both *age* and *gender* cause bias *(hb_+age_+gender)*. The plot illustrates kernel density estimates of accuracy, with shaded regions representing the distribution spread.

To assess fairness, we measure ΔDP at the decision point following *RELEASE*, where *CODE OK* is considered the positive outcome. Since *gender* includes three categories, we evaluate ΔDP for the split into *gender-conforming* individuals (*male* and *female*) and *non-conforming* individuals. For *age*, we split the group into patients are 85 or older versus those younger than 85.

5.4 BPI Challenge 2012

The **BPI Challenge 2012** [Don12] event log, consisting of 13,087 cases, is a well-known dataset in process mining, sourced from a Dutch financial institution. It captures multiple subprocesses within a loan application procedure, each distinguished by specific prefixes. These subprocesses include application states, denoted by the *A_* prefix, offer states, represented by the *O_* prefix, and work items, labeled with the *W_* prefix.

Event Log	Accuracy		
	Base	Modified	Enriched
-age, -gender	.915 ± .003	.916 ± .002	.916 ± .002
-age, +gender	.915 ± .003	.921 ± .002	.926 ± .002
+age, -gender	.915 ± .003	.918 ± .005	.928 ± .005
+age, +gender	.915 ± .003	.924 ± .003	.932 ± .002

Event Log	Demographic Parity (Age)			Demographic Parity (Gender)		
	Base	Modified	Enriched	Base	Modified	Enriched
-age, -gender	.007 ± .004	.006 ± .002	.006 ± .002	.007 ± .004	.013 ± .014	.013 ± .014
-age, +gender	.009 ± .005	.014 ± .022	.044 ± .008	.014 ± .013	.110 ± .168	.840 ± .036
+age, -gender	.012 ± .011	.015 ± .023	.891 ± .023	.006 ± .002	.006 ± .004	.084 ± .029
+age, +gender	.015 ± .011	.068 ± .119	.921 ± .028	.012 ± .010	.069 ± .122	.832 ± .039

Tab. 5.1: Evaluation of accuracy and demographic parity for the four versions of the *Hospital Billing* log. The attributes *age* and *gender* are annotated based on whether they introduce a bias (+) or not (-). The reported values represent the mean (μ) and standard deviation (σ) across validation folds, expressed as $\mu \pm \sigma$.

For our analysis, we focus specifically on the *A_* subprocess, which pertains to the application phase of the loan procedure. This subprocess is particularly relevant for examining fairness concerns, as it involves crucial decision points that influence applicants' outcomes. Within this subprocess, there are 10 distinct event types, capturing key activities such as application submission, assessment, approval, and rejection. Although the event log doesn't contain any sensitive case attributes, the attribute *AMOUNT_REQ* specifies the amount of loan requested by the customer. Since the amount is crucial information for deciding whether an application should be accepted, we consider it to be a non-sensitive case attribute, which can also be utilized by the fair Base Model.

In order to explore our fairness problem statement, we enrich the event log with the sensitive categorical case attribute *gender*. To explore fairness concerns, we enrich the event log with the categorical sensitive case attribute *gender*. The DFG highlights two key decision points where biases could be introduced:

- Whether an application transitions to *A_PREACCEPTED* or *A_DECLINED* after *A_PARTLYSUBMITTED*.

- Which activity follows after *A_FINALIZED*, with the most common options being either *A_APPROVED* or *A_CANCELLED*.

Therefore, we choose the assigment rules for *gender* are as follows:

- **If** the event sequence contains *A_APPROVED*, **then** *female*: 70%, *male*: 30%.

- **Else if** the event sequence contains *A_CANCELLED*, **then** *female*: 30%, *male*: 70%.

- **Else if** the event sequence contains *A_PARTLYSUBMITTED* followed by *A_PREACCEPTED*, **then** *female*: 70%, *male*: 30%.

- **Else if** the event sequence contains *A_PARTLYSUBMITTED* followed by *A_DECLINED*, **then** *female*: 30%, *male*: 70%.

- **Else** (if none of the above conditions apply), **then** *female*: 50%, *male*: 50%.

These rules result in around 56.6% male and 43.4% female applicants.

In the context of this event log, we only consider demographic disparities at decision of whether an application transitions to *A_PREACCEPTED* or *A_DECLINED* to be unfair. Thus, we measure ΔDP for samples with the previous activity *A_PARTLYSUBMITTED*, considering *A_PREACCEPTED* as the positive outcome. Since *gender* is binary in this dataset, we compare its impact on the probability of *A_PREACCEPTED* between *male* and *female* applicants.

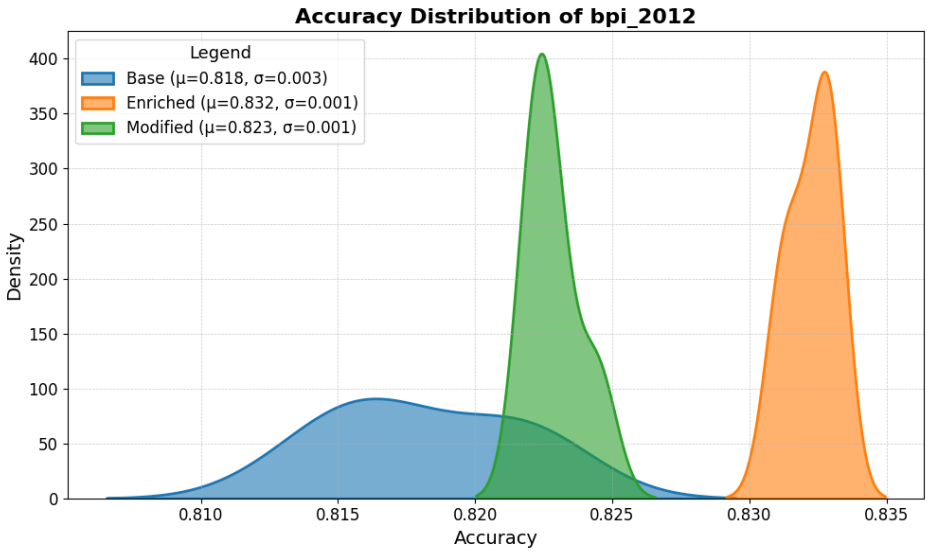

Fig. 5.5: Distribution of accuracy values for the base, enriched, and modified models on the *BPI Challenge 2012 (bpi_2012)* event log. The plot illustrates kernel density estimates of accuracy, with shaded regions representing the distribution spread.

5.5 Discussion

Event Log	Accuracy			Demographic Parity		
	Base	Modified	Enriched	Base	Modified	Enriched
cs	.821 ± .004	.850 ± .003	.895 ± .002	.001 ± .001	.013 ± .024	.999 ± .002
hb_gender	.915 ± .003	.921 ± .002	.926 ± .002	.014 ± .013	.110 ± .168	.840 ± .036
hb_age	.915 ± .003	.918 ± .005	.928 ± .005	.012 ± .011	.015 ± .023	.891 ± .023
bpi_2012	.818 ± .003	.823 ± .001	.832 ± .001	.058 ± .019	.054 ± .014	.525 ± .036

Tab. 5.2: Evaluation of accuracy and demographic parity for the *cancer screening* log (cs), the *BPI Challgenge 2012* log (bpi_2012) and two versions of the *Hospital Billing* log, where only the attribute *age* introduces a bias (hb_age) and only the attribute *gender* introduces a bias (hb_gender). The reported values represent the mean (μ) and standard deviation (σ) across validation folds, expressed as $\mu \pm \sigma$.

6 Ablation Study

In this chapter, we investigate the individual contributions of key factors influencing the performance and behavior of our proposed method. Specifically, we focus on the effects of bias strength, the number of sensitive attributes, and the number of decisions on the system's overall functionality. These factors are central to understanding the robustness and scalability of our approach.

6.1 Bias Strength

First, we will evaluate the sensitivity of the method to varying levels of inherent bias in the training data. For this experiment, we used the *cancer screening* model as a basis, due to its easily adjustable parameters. While the model remains largely consistent with the version described in section 5.2, we modify the bias strength for the attribute *gender* at the gateway following the activity *assess screening eligibility*, which determines transitions to *collect patient history* and *refuse screening*. Before, the probabilities to transition to either of these activities were split 0.7 to 0.3, depending on whether the patient is *male* or *female*. In this analysis, we vary the probabilities to transition to *collect patient history* for *male* patients and *refuse screening* for *female* patients, incrementally from 0.5 (perfectly equal) to 1.0 (entirely biased) in steps of 0.05. The rest of the model remains identical.

For each step in bias strength and each of the three models (base, enriched and modified), we perform 5-fold cross-validation and calculate the mean and standard deviation for both accuracy and demographic parity. Analogously to the evalution of the *cancer screening* model, the demographic parity computation is performed by measuring the difference in selection rates between male and female patients for the *collect history* transition, using only samples where the previous event was *assess eligibility*. Therefore, when modifying the distilled decision tree, we remove each node that uses *gender* as its splitting feature and contains leaf nodes with the output classes *collect patient history* and *refuse screening* in its subtree. The node removal is done via the retraining method. The corresponding results are presented in Table 6.1 and Figure 6.1.

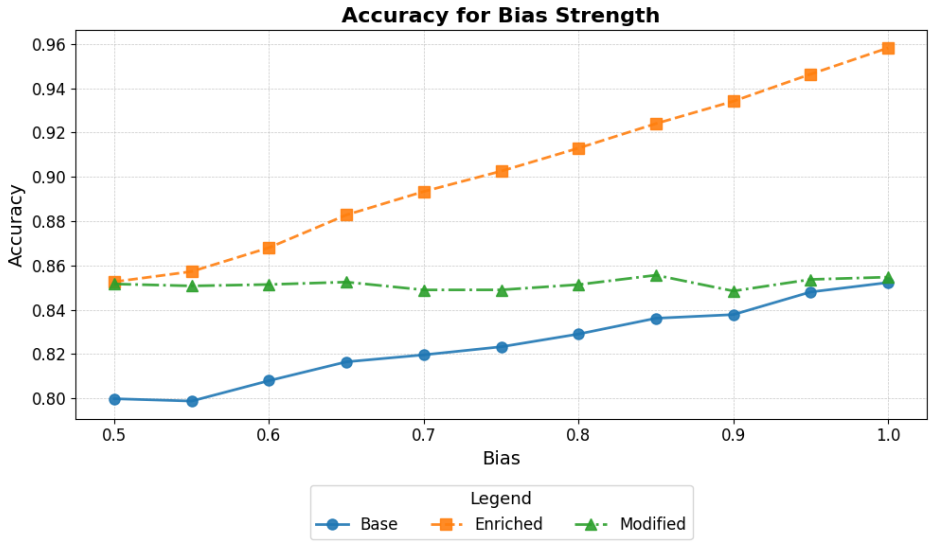

Fig. 6.1: Impact of varying bias strength on the accuracy. Each point in the graph represents the mean accuracy across validation folds for a certain bias strength.

Bias Strength	Accuracy			Demographic Parity		
	Base	Modified	Enriched	Base	Modified	Enriched
0.50	.800 ± .001	.852 ± .002	.853 ± .003	.002 ± .001	.111 ± .222	.116 ± .231
0.55	.799 ± .005	.851 ± .003	.857 ± .006	.000 ± .000	.020 ± .035	.597 ± .487
0.60	.808 ± .003	.851 ± .002	.868 ± .004	.000 ± .001	.014 ± .022	.903 ± .183
0.65	.816 ± .002	.852 ± .002	.883 ± .003	.002 ± .001	.008 ± .014	.997 ± .002
0.70	.820 ± .003	.849 ± .003	.893 ± .002	.001 ± .001	.004 ± .006	.996 ± .002
0.75	.823 ± .003	.849 ± .003	.903 ± .002	.001 ± .002	.000 ± .000	.998 ± .002
0.80	.829 ± .004	.851 ± .002	.913 ± .003	.002 ± .002	.003 ± .001	.998 ± .001
0.85	.836 ± .002	.856 ± .003	.924 ± .001	.005 ± .003	.045 ± .038	.999 ± .000
0.90	.838 ± .005	.848 ± .004	.934 ± .002	.004 ± .003	.001 ± .000	1.00 ± .000
0.95	.848 ± .002	.854 ± .001	.946 ± .001	.006 ± .004	.003 ± .002	1.00 ± .001
1.00	.852 ± .003	.855 ± .002	.958 ± .001	.004 ± .002	.007 ± .008	1.00 ± .000

Tab. 6.1: Evaluation of accuracy and demographic parity for varying bias strengths. The reported values represent the mean (μ) and standard deviation (σ) across validation folds, expressed as $\mu \pm \sigma$.

6.2 Number of Sensitive Attributes

Next, we will examine the behavior of our approach when varying the number of sensitive attributes for a single decision point. For this purpose, we again use a modified version of the *cancer screening* model described in section 5.2. In this modified model, we remove the case attribute *gender* and introduce n generic categorical attributes a_i, with $i \in \{1, ..., n\}$. Each of these attributes is considered to be sensitive and can take one of two possible values, A or B, with a chance of 50% each.

In a similar manner to the original *gender* attribute, these attributes a_i influence the same two decision gateways. However, since the transition probability at these gateways can no longer be determined directly by the value of a single attribute, we follow an approach where each attribute contributes a weight that allows us to calculate the appropriate transition probabilities. The probability of selecting a particular transition is computed by multiplying the individual attribute weights, normalized by the sum of the multiplied weights for all possible transitions.

The unfair bias, which we aim to mitigate in the *cancer screening* model, still occurs at the gateway after *assess screening eligibility*. The transition leading to *collect patient history* is assigned a weight of 0.7 by each generic attribute with the value A and 0.3 by each attribute with the value B. Conversely, the transition leading to *refuse screening* is given a weight of 0.3 when the attribute value is A and 0.7 when it is B. For example, if we consider three attributes a_1, a_2, a_3 with the attributes values A, B, A, the product of weights would be $0.7 \times 0.3 \times 0.7 = 0.147$ for *collect patient history*, and $0.3 \times 0.7 \times 0.3 = 0.063$ for *refuse screening*. Consequently, the resulting transition probabilities are $\frac{0.147}{0.147+0.063} = 0.7$ for *collect patient history* and $\frac{0.063}{0.147+0.063} = 0.3$ for *refuse screening*.

The attributes a_i also effect the gateway after *collect patient history*, when deciding whether an individual should proceed to *prostate screening* or *mammary screening*, representing bias that we consider to be acceptable. When assigning the corresponding weights for the possible transitions, we introduce a dependency on the index of the attribute. If the index i of an attribute a_i is even, the transition to *collect patient history* is assigned a weight of 0.7 when the attribute takes the value A and 0.3 when it takes the value B, while the transition to *refuse screening* is given the weight of 0.3 for A and 0.7 for B. However, when the index i is odd, these weight assigments are reversed for the values A and B. This is done to circumvent the effect observed in the previous ablation study from section 6.1, where an increased selection rate for *collect patient history* automatically leads to an increased selection rate for *prostate screening*.

We evaluate the model by varying the number of attributes up to 10 in increments of 2. The demographic parity computation is performed by measuring the difference in selection rates between attribute values A and B for the *collect history* transition, averaged over all attributes a_i, using only samples where the previous event was *assess eligibility*. Similar to the ablation study for bias strength, when modifying the distilled decision tree, we remove each node that uses an attribute a_i as its splitting feature and contains leaf nodes with the output classes *collect patient history* and *refuse screening* in its subtree. Due to the high amount of potentially biasing attributes, utilizing the retraining method when removing nodes would likely introduce new unwanted biases in the retrained subtree, requiring many iterations of

retraining in an already long evaluation process. Therefore, we applied the method of cutting branches instead. Again, we employ 5-fold cross-validation for each increment and model. The results are reported in Table 6.2 and Figure 6.2.

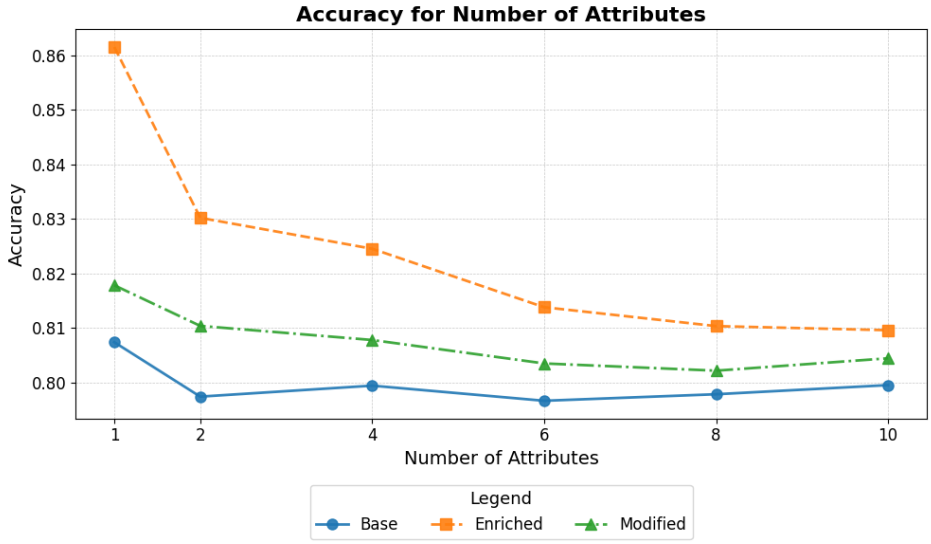

Fig. 6.2: Impact of varying number of sensitive attributes on the accuracy. Each point in the graph represents the mean accuracy across validation folds for a certain number of attributes.

Num. Attributes	Accuracy			Demographic Parity		
	Base	Modified	Enriched	Base	Modified	Enriched
1	$.807 \pm .003$	$.818 \pm .003$	$.862 \pm .005$	$.000 \pm .000$	$.001 \pm .000$	$.998 \pm .000$
2	$.797 \pm .005$	$.810 \pm .004$	$.830 \pm .003$	$.001 \pm .001$	$.004 \pm .001$	$.495 \pm .067$
4	$.799 \pm .003$	$.808 \pm .005$	$.825 \pm .004$	$.000 \pm .000$	$.009 \pm .000$	$.376 \pm .038$
6	$.797 \pm .003$	$.803 \pm .007$	$.814 \pm .006$	$.001 \pm .000$	$.010 \pm .004$	$.273 \pm .027$
8	$.798 \pm .003$	$.802 \pm .002$	$.810 \pm .005$	$.001 \pm .000$	$.033 \pm .014$	$.211 \pm .039$
10	$.799 \pm .003$	$.804 \pm .002$	$.810 \pm .003$	$.003 \pm .002$	$.033 \pm .010$	$.157 \pm .051$

Tab. 6.2: Evaluation of accuracy and demographic parity for a varying amount of sensitive attributes. The reported values represent the mean (μ) and standard deviation (σ) across validation folds, expressed as $\mu \pm \sigma$.

6.3 Number of biased Decisions

In this section, we examine the effect of varying the number of decision points that utilize sensitive attributes in the model. Similar to the previous ablation study on the number of sensitive attributes, this study aims to assess the scalability of our method when multiple decisions in a process model introduce biased behavior.

The models and event logs considered so far do not contain a sufficient number of meaningful decision points to conduct this analysis. Therefore, we use a deliberately constructed process model that allows for a controlled evaluation of our approach.

This model consists of a *start* activity and an *end* activity. Between these, it contains n groups of four activities, denoted as A_i, B_i, C_i, and D_i, where $i \in \{1, ..., n\}$. Each group represents a decision structure in which an incoming case must follow one of two possible paths, determined by a sensitive attribute. Similar to the previous experiment, we introduce n sensitive categorical attributes a_i, where each a_i takes the value A or B with an equal probability of 50%. Each attribute a_i affects two decision gateways within its respective group.

The first decision occurs when entering the i-th group, where the transition probabilities are biased as follows: If $a_i = A$, the probability of selecting A_i is 0.7, while the probability of selecting B_i is 0.3. Conversely, if $a_i = B$, the probability of selecting A_i is 0.3, and the probability of selecting B_i is 0.7. These decisions, concerning the events A_i and B_i, introduce the unfair bias that we aim to remove in our model.

The second decision occurs after A_i or B_i, determining whether the process continues to C_i or D_i. This decision is structured in a similar way: If $a_i = A$, the probability of selecting C_i is 0.7, while the probability of selecting D_i is 0.3. If $a_i = B$, the probabilities are reversed, with C_i receiving a probability of 0.3 and D_i a probability of 0.7. The biases in these decisions, concerning the events C_i and D_i, are considered to be acceptable. As such, our approach should retain these biases while mitigating the unfair bias at the first decision point in each group.

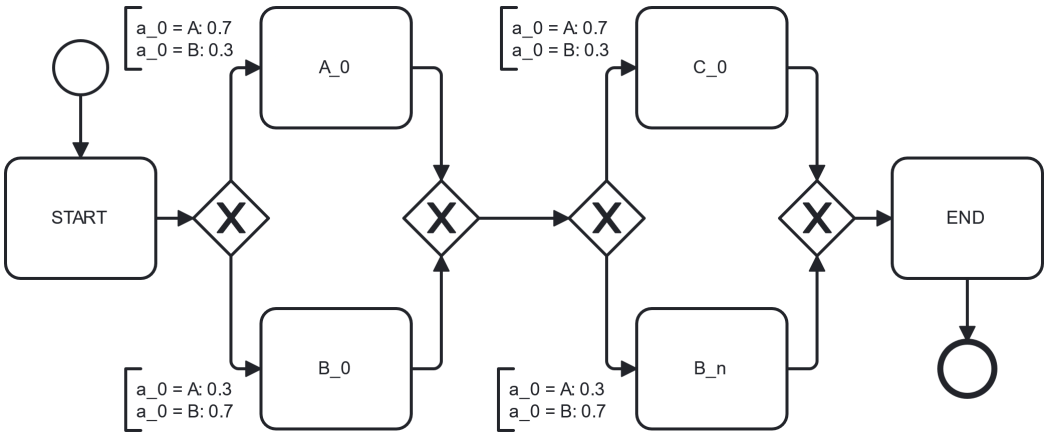

Fig. 6.3: The process model with only a single group consisting of the activities A_0, B_0, C_0 and D_0, influenced by the attribute a_0. Additional groups would be added between the last gateway, after activities C_0 and D_0, and the *end* activity.

Since each of the n groups introduces two biased decision points, a model containing n groups corresponds to a model with $2n$ biased decisions. We inspect the number of biased decisions up to 20, in increments of 4. When mitigating the unfair bias in the distilled decision tree, we remove each node that uses an attribute a_i as its splitting feature and contains A_i and B_i as leaf nodes within its subtree. As before, given the high number of biased decisions, applying the retraining method would

introduce additional unwanted biases in the retrained subtree. Therefore, we again use the method of cutting branches for node removal. The results from applying 5-fold cross-validation for each increment and model are reported in Table 6.3 and Figure 6.4.

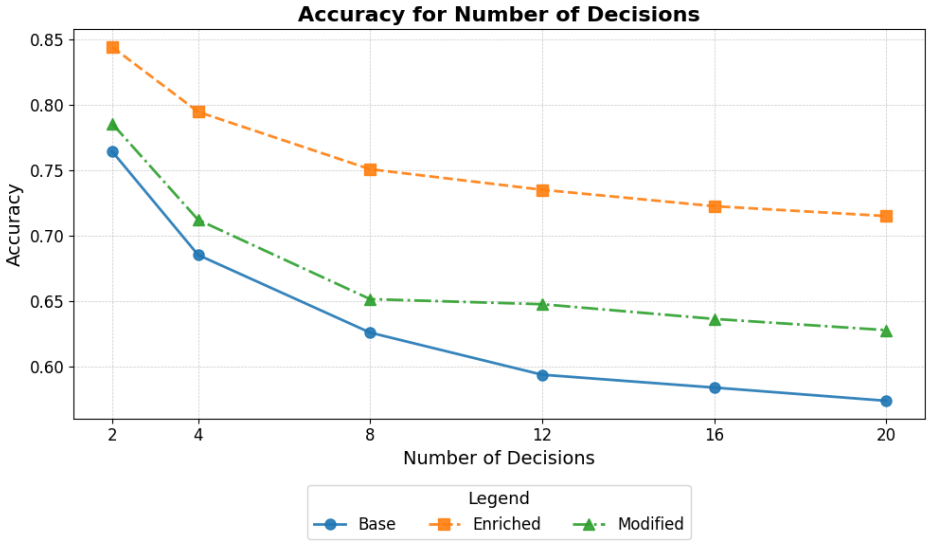

Fig. 6.4: Impact of varying number of biased decisions on the accuracy. Each point in the graph represents the mean accuracy across validation folds for a certain number of decisions.

Num. Decisions	Accuracy			Demographic Parity		
	Base	Modified	Enriched	Base	Modified	Enriched
2	$.764 \pm .003$	$.786 \pm .017$	$.844 \pm .004$	$.001 \pm .000$	$.002 \pm .000$	$.997 \pm .000$
4	$.685 \pm .005$	$.712 \pm .018$	$.795 \pm .002$	$.002 \pm .002$	$.011 \pm .008$	$.997 \pm .003$
8	$.626 \pm .002$	$.652 \pm .015$	$.751 \pm .002$	$.001 \pm .001$	$.004 \pm .003$	$.995 \pm .002$
12	$.594 \pm .004$	$.648 \pm .002$	$.735 \pm .001$	$.001 \pm .001$	$.006 \pm .006$	$.992 \pm .008$
16	$.584 \pm .002$	$.636 \pm .002$	$.723 \pm .001$	$.005 \pm .005$	$.003 \pm .004$	$.978 \pm .014$
R	$.574 \pm .003$	$.628 \pm .002$	$.715 \pm .001$	$.004 \pm .004$	$.002 \pm .003$	$.977 \pm .009$

Tab. 6.3: Evaluation of accuracy and demographic parity for a varying amount of biased decisions. The reported values represent the mean (μ) and standard deviation (σ) across validation folds, expressed as $\mu \pm \sigma$.

6.4 Discussion

7 Conclusions

This chapter provides an overview of the key implications derived from our results, discusses the limitations of our approach, and outlines potential directions for future research.

7.1 Implications for theory and practise

One important theoretical implication of our work is the recognition that not all forms of bias in machine learning models should be considered inherently negative. While bias is often viewed as a flaw that must be eliminated, certain biases can reflect domain-specific requirements that contribute to more accurate and context-sensitive predictions. Our experiments have demonstrated that it is possible to remove unwanted biases while preserving beneficial ones, highlighting the need for a more nuanced understanding of bias in machine learning.

From a practical perspective, ensuring fairness in machine learning models is increasingly important, especially in light of legal and regulatory frameworks such as the European Union's Artificial Intelligence Act [Com21]. These regulations emphasize the need for machine learning systems to be transparent, unbiased, and equitable in their decision-making processes. Our method offers a practical way to incorporate fairness into the model development pipeline, allowing for both transparent evaluation and targeted adjustments to meet fairness standards, without significantly compromising accuracy.

Even in scenarios where fairness-driven changes to the model are not strictly necessary, the insights gained from analyzing and addressing bias can play a critical role in building trust among stakeholders. Understanding how a black-box model behaves and ensuring it aligns with ethical and legal standards can foster confidence in its use, particularly in socially sensitive applications such as hiring, healthcare, and law enforcement. Thus, our fairness-related insights would benefit not only those directly affected by the model's decisions but also the broader ecosystem of users, developers, and regulators.

7.2 Limitations and Challenges

While our approach shows considerable promise, several limitations and challenges became evident during the course of this work.

Decision Tree Splits

The success of our methods heavily depends on how the splits in the distilled decision tree are structured. To effectively remove negative bias associated with a sensitive attribute while preserving positive bias, it is crucial for these biases to manifest in separate splits. If both types of biases are merged into a single node, they become impossible to separate using our approach. This issue is more likely to occur in higher-level splits (nodes with low depth), as early splits are more general and less context-specific compared to deeper splits.

A potential approach to address this is to use a modified training algorithm that penalizes splits based on sensitive attributes, similar as proposed in [KCP10]. This would cause sensitive features to appear at deeper-level splits (nodes with higher depth), where they can be evaluated in a more context-sensitive manner. However, this approach may come at the cost of accuracy and transparency, as the decision tree might no longer directly capture the inner workings of its teacher model.

Decision Tree Accuracy

Decision trees may struggle to handle complex tasks with the same effectiveness as deep learning methods. As a result, the performance gap between the distilled decision tree and its teacher model can widen significantly as task complexity increases. This issue is worsened by the decision tree's reliance on discrete target labels, which can be too coarse to capture the nuances of the model's output distribution. Consequently, the decision tree may fail to fully leverage the wealth of information contained in the model's predictions, leading to significantly lower accuracies.

One potential solution is to use soft decision trees as the student model, as proposed in [FH17]. Soft decision trees employ probabilistic decision boundaries, enabling them to better incorporate the output distribution from the teacher model. While they do retain a level of interpretability, as decisions are still based on hierarchical rules, the probabilistic nature of decisions can make them less straightforward than hard decision rules.

Even with this potential improvement, the accuracy of the decision tree remains a critical factor in fine-tuning. If the tree generates inaccurate relabeling, it could hinder the fine-tuned model's performance. To address this, one could implement early stopping during fine-tuning, helping to balance fairness improvements with accuracy retention and avoiding overfitting to imprecise relabeling.

Decision Tree Complexity

As task complexity increases, the complexity of the decision tree also rises. With more activities, attributes, and dependencies, more splits and nodes are required in the decision tree to capture these intricacies, resulting in a larger, more intricate model that becomes increasingly difficult for domain experts to interpret.

Limiting the number of nodes and tree depth can preserve interpretability, but this may come at the cost of accuracy, as the tree may no longer represent the internal processes of the neural network to the full extent. This tradeoff between accuracy and interpretability seems to be an inherent bottleneck of our approach, especially for highly complex problems.

Manual Labor

Finally, compared to methods from related work such as [QA19], [LP24], or [PDV24], which aim to achieve fairness automatically based on predefined rules, our approach requires more direct input from domain experts. While it is possible to use our method automatically to eliminate all bias from a sensitive attribute by removing all nodes that use it for splitting, preserving positive biases requires domain experts to carefully examine and adjust the decision tree. In practice, this means that our approach is likely to be more labor-intensive and consequently more cost-intensive than comparable methods.

7.3 Future Work

Besides addressing the challenges we listed in the previous section 7.2, future research could expand on this thesis in several directions.

Event Attributes

Since this thesis focused exclusively on case attributes, future work could expand by incorporating event attributes to further enhance both fairness analysis and model performance. These event attributes could be encoded in a similar manner to case attributes and integrated into the input vector for each recent event within the time window considered. Furthermore, including timestamp data, such as time deltas for each event rather than just the most recent one, could provide a richer temporal context and potentially improve the predictive capabilities of the models as well.

Fairness Metrics

Our evaluation exclusively utilized demographic parity as a fairness metric, which, while straightforward, can be imprecise for numerical data due to its dependence on predefined thresholds. Future studies could explore threshold-independent metrics, as utilized in [PDV24], which better capture fairness across continuous attributes.

Other approaches to group fairness, such as Shapley values [Sha53] as used in [LP24], are worth investigating due to their ability to provide a attribution-based perspective on bias. Unlike demographic parity, which evaluates fairness at a group level by comparing outcome distributions across demographic groups, Shapley values assess the contribution of individual features to a model's decisions, which allows for pinpointing the specific attributes that drive disparities in predictions.

Additionally, future work could examine how the proposed methodology influences individual fairness metrics, Unlike group fairness, which focuses on comparing outcomes across demographic groups, individual fairness emphasizes treating comparable individuals within demographic groups similarly.

Prediction Tasks

The proposed methodology could be extended to other key tasks in PBPM, such as outcome prediction and remaining-time prediction, to evaluate its broader applicability.

Outcome prediction involves determining the final result of a process, such as whether a loan will be approved or a treatment will succeed. Like next-activity prediction, outcome prediction is a classification task, making it likely that our proposed methodology could be applied with minimal adjustments: Simply replacing

the next activity with the process outcome as the target labels in the dataset might suffice.

Remaining-time prediction, however, introduces additional challenges. As a regression problem, it requires a fairness metric that is suitable for continuous outputs, such as the Shapley values [Sha53], which were used for evaluating the remaining time task in [LP24]. Moreover, since traditional decision trees are not designed for continuous outputs, an appropriate white-box model, such as regression trees [Cor+09], would need to be used as the student model for knowledge distillation.

Correlations

Fairness and debiasing research frequently highlights the issue of correlated attributes, as simply removing access to a sensitive attribute does not guarantee the elimination of its influence if it correlates with other non-sensitive attributes used in decision-making.

For instance, in the BPI Challenge 2012 dataset discussed in section **??**, the event log includes the non-sensitive attribute *amount_req*, which represents the amount of money requested from the lending firm. Higher request amounts typically warrant additional scrutiny, making it reasonable for inquiries with larger sums to be denied more often. However, if *amount_req* is correlated with a sensitive attribute like *gender*, for example, if women generally request higher sums than men, a "fair" model that relies on this attribute might unintentionally deny requests from women more frequently, even without direct access to the attribute *gender*.

Determining whether such correlations are fair requires careful consideration on a case-by-case basis, as some correlations may be justified while others may perpetuate bias. To adress this, future research should investigate how correlations involving sensitive attributes may appear in the distilled decision trees and develop methods to communicate information about such correlations to end-users in a clear and transparent manner.

Alternative Classifiers

The proposed methodology is not inherently tied to neural networks and could potentially be applied in a model-agnostic manner. Future research should explore the impact of using different black-box models, such as transformer-based architectures [BSD21], to assess whether the approach generalizes well across different classifier types.

Comparisons to Related Work

In this thesis, we evaluated our approach solely against a fully fair and a fully biased model. However, future work should include comparisons with alternative fairness methods in PBPM, such as those proposed in [QA19], [LP24], and [PDV24]. These comparisons may necessitate adjustments to our pipeline to account for differences in datasets, prediction tasks, or evaluation metrics used by these other approaches. Nonetheless, such comparisons would offer a more comprehensive view of the relative strengths and weaknesses of various methods, helping to position our approach within the broader landscape of fairness in PBPM.

Bibliography

[Bre17] Leo Breiman. *Classification and regression trees*. Routledge, 2017 (cit. on pp. 14, 16, 17).

[BSD21] Zaharah A Bukhsh, Aaqib Saeed, and Remco M Dijkman. „Processtransformer: Predictive business process monitoring with transformer network“. In: *arXiv preprint arXiv:2104.00721* (2021) (cit. on p. 51).

[CDGR19] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. „Learning accurate LSTM models of business processes“. In: *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*. Springer. 2019, pp. 286–302 (cit. on p. 4).

[CH24] Simon Caton and Christian Haas. „Fairness in machine learning: A survey“. In: *ACM Computing Surveys* 56.7 (2024), pp. 1–38 (cit. on p. 3).

[Cha23] Xinyu Chang. „Gender Bias in Hiring: An Analysis of the Impact of Amazon's Recruiting Algorithm“. In: *Advances in Economics, Management and Political Sciences* 23 (2023), pp. 134–140 (cit. on p. 1).

[Cho15] François Chollet. *Keras*. `https://github.com/fchollet/keras`. 2015 (cit. on p. 25).

[Com21] European Commission. „Proposal for a Regulation Laying Down Harmonized Rules on Artificial Intelligence (Artificial Intelligence Act)“. In: *COM/2021/206 final* (2021) (cit. on p. 47).

[Cor+09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd. MIT Press, 2009 (cit. on pp. 4, 14, 51).

[DAFST22] Maria De-Arteaga, Stefan Feuerriegel, and Maytal Saar-Tsechansky. „Algorithmic fairness in business analytics: Directions for research and practice“. In: *Production and Operations Management* 31.10 (2022), pp. 3749–3770 (cit. on p. 3).

[Don12] Boudewijn van Dongen. *BPI Challenge 2012*. `https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`. Version 1. dataset. 2012 (cit. on pp. 4, 36).

[Dwo+12] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. „Fairness Through Awareness“. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM. 2012, pp. 214–226 (cit. on pp. 4, 19).

[FH17] Nicholas Frosst and Geoffrey Hinton. „Distilling a neural network into a soft decision tree“. In: *arXiv preprint arXiv:1711.09784* (2017) (cit. on p. 48).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. „Generative adversarial nets“. In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 3).

[Han+23] Xiaotian Han, Zhimeng Jiang, Hongye Jin, et al. „Retiring Delta DP: New Distribution-Level Metrics for Demographic Parity“. In: *arXiv preprint arXiv:2301.13443* (2023) (cit. on p. 4).

[HPS16] Moritz Hardt, Eric Price, and Nati Srebro. „Equality of opportunity in supervised learning“. In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 4).

[HRW86] Geoffrey E. Hinton, David E. Rumelhart, and Ronald J. Williams. „Learning representations by back-propagating errors“. In: *Nature* 323 (1986), pp. 533–536 (cit. on p. 4).

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. „Distilling the Knowledge in a Neural Network“. In: *arXiv preprint arXiv:1503.02531* (2015) (cit. on p. 26).

[KCP10] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. „Discrimination Aware Decision Tree Learning“. In: *2010 IEEE International Conference on Data Mining*. 2010, pp. 869–874 (cit. on p. 48).

[Kin14] Diederik P Kingma. „Adam: A method for stochastic optimization“. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 11).

[Koh95] R Kohavi. „A study of cross-validation and bootstrap for accuracy estimation and model selection“. In: *Morgan Kaufman Publishing* (1995) (cit. on p. 32).

[Kra+21] Wolfgang Kratsch, Jonas Manderscheid, Maximilian Röglinger, and Johannes Seyfried. „Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction“. In: *Business & Information Systems Engineering* 63 (2021), pp. 261–276 (cit. on p. 29).

[LP24] Massimiliano de Leoni and Alessandro Padella. „Achieving Fairness in Predictive Process Analytics via Adversarial Learning (Extended Version)“. In: *arXiv preprint arXiv:2410.02618* (2024) (cit. on pp. 3, 4, 8, 49–52).

[LWM18] Xuan Liu, Xiaoguang Wang, and Stan Matwin. „Improving the interpretability of deep neural networks with knowledge distillation“. In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 905–912 (cit. on p. 26).

[Man17] Felix Mannhardt. *Hospital Billing - Event Log*. `https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741`. Version 1. dataset. 2017 (cit. on pp. 4, 24, 34).

[PDV24] Jari Peeperkorn and Simon De Vos. „Achieving Group Fairness through Independence in Predictive Process Monitoring“. In: *arXiv preprint arXiv:2412.04914* (2024) (cit. on pp. 4, 49, 50, 52).

[Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 26).

[Poh+23] Timo Pohl, Alessandro Berti, Mahnaz Sadat Qafari, and Wil MP van der Aalst. „A Collection of Simulated Event Logs for Fairness Assessment in Process Mining“. In: *arXiv preprint arXiv:2306.11453* (2023) (cit. on pp. 4, 21).

[PY09] Sinno Jialin Pan and Qiang Yang. „A Survey on Transfer Learning“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2009), pp. 1345–1359 (cit. on p. 29).

[PZ19] Victor M Panaretos and Yoav Zemel. „Statistical aspects of Wasserstein distances“. In: *Annual review of statistics and its application* 6.1 (2019), pp. 405–431 (cit. on p. 4).

[QA19] Mahnaz Sadat Qafari and Wil Van der Aalst. „Fairness-aware process mining“. In: *On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21–25, 2019, Proceedings*. Springer. 2019, pp. 182–192 (cit. on pp. 3, 4, 14, 19, 34, 49, 52).

[RMVL21] Efrén Rama-Maneiro, Juan C Vidal, and Manuel Lama. „Deep learning for predictive business process monitoring: Review and benchmark“. In: *IEEE Transactions on Services Computing* 16.1 (2021), pp. 739–756 (cit. on p. 8).

[Rud19] Cynthia Rudin. „Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead“. In: *Nature machine intelligence* 1.5 (2019), pp. 206–215 (cit. on p. 8).

[Sha53] Lloyd S Shapley. „A value for n-person games“. In: *Contribution to the Theory of Games* 2 (1953) (cit. on pp. 4, 50, 51).

[WDW21] Hans Weytjens and Jochen De Weerdt. „Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring“. In: *International Conference on Business Process Management*. Springer. 2021, pp. 18–29 (cit. on p. 23).

[Whi04] Stephen A White. „Introduction to BPMN“. In: *Ibm Cooperation* 2.0 (2004), p. 0 (cit. on p. 5).

List of Figures

List of Tables

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

Bayreuth, August 26, 2015

Vorname Nachname